Corso di Laurea in Sicurezza dei Sistemi e delle Reti Informatiche

Università Statale di Milano
Anno Accademico 2017/2018

Esame di Programmazione
Relazione sullo sviluppo del gioco “Forza 4” in C

Albino Salvatore

Sommario

1. Introduzione 3
2. Descrizione delle funzionalità del programma 3
2.1 Caso 1: vittoria di uno dei giocatori 4
2.2 Caso 2: Partita terminata in parità 4
3. Composizione del programma 5
3.1 Descrizione del file forza4.c 5
3.2 Descrizione del file campogioco.h 7
3.3 Descrizione del file campogioco.c 7
3.4 Descrizione del file stampe.h 8
3.5 Descrizione del file stampe.c 8

1. Introduzione

Il programma sviluppato per l'esame ha lo scopo di simulare il famoso gioco da tavola "Forza 4", nel quale due giocatori si sfidano cercando di allineare 4 delle proprie pedine all'interno di un tabellone. Al termine della partita
Il linguaggio utilizzato per la realizzazione è il C, mentre l'ambiente di sviluppo utilizzato è stato eclipse-cpp.
Il codice sorgente del programma è disponibile anche online su github all'indirizzo https://github.com/nowhereb0y/forza4 .

2. Descrizione delle funzionalità del programma

Il programma permette a due giocatori di sfidarsi l'uno contro l'altro, permettendo di scegliere se effettuare una rivincita dopo ogni partita.
Il campo da gioco è rappresentato da una matrice di n righe ed m colonne, impostate rispettivamente a 6 e 7 mediante delle costanti definite in un file header. La scelta di questi valori è stata fatta per mantenere le stesse dimensioni presenti nel gioco originale.
Una volta avviato il programma, verrà mostrato un banner realizzato con caratteri ascii insieme ad una spiegazione del gioco dopodichè ai giocatori verrà chiesto di inserire il proprio nome. Tale informazione verrà memorizzata in una variabile di tipo strutturata che conterrà, oltre al nome, il simbolo che sarà utilizzato come pedina, e il numero di vittorie che il giocatore riuscirà ad ottenere nel corso della sessione di gioco.
Una volta superata questa fase si entra in un ciclo while con una variabile a guardia dell'esecuzione. La partita sarà iniziata dal giocatore che per primo ha inserito il proprio nome, nel caso di rivincita l'ordine di partenza sarà di volta in volta invertito.
I giocatori si alterneranno nel gioco scegliendo una colonna dove inserire la pedina. Questa informazione viene chiesta all'utente tramite una funzione che ha anche il compito di verificare che l'input dell'utente sia corretto (viene controllato che la colonna non sia già piena e che il valore inserito corrisponda a una colonna esistente). L'output di questa funzione viene poi passata ad una procedura che si occupa di inserire il simbolo del giocatore all'interno della matrice. Terminata questa procedura viene chiamata una funzione che verifica se ci sono 4 pedine allineate in orizzontale, verticale o diagonale.

2.1 Caso 1: vittoria di uno dei giocatori

Nel caso in cui vengano rilevate 4 pedine allineate, vengono effettuate le seguenti azioni:Il contatore delle vittorie contenuto all'interno della variabile strutturata del giocatore viene incrementato, così come il contatore delle partite (tale variabile di tipo intero è utilizzata dalla procedura che gestisce la rivincita per fare in modo che i giocatori si alternino nell'iniziare la partita). Successivamente viene chiamata una procedura che stampa a video il numero di partite vinte da ogni giocatore ed il numero di partite terminate in parità dopodiché viene chiamata una funzione che chiede all'utente se desidera effettuare una nuova partita oppure se vuole uscire dal programma. Nel caso di una nuova partita, viene impostato opportunamente il contatore delle mosse in modo da far iniziare per primo il giocatore che nella precedente partita aveva iniziato per secondo, il campo di gioco viene inizializzato sovrascrivendo i caratteri esistenti con degli spazi mentre la variabile a guardia del ciclo while, che era stata modificata in precedenza in fase di controllo della vittoria viene reimpostata in modo da poter ripartire con il ciclo.

2.2 Caso 2: Partita terminata in parità

All'interno del ciclo principale, è presente una if che verifica il numero delle mosse effettuate: quando il contatore delle mosse raggiunge un valore pari al totale delle caselle disponibili nella matrice e nessun utente è riuscito ad allineare 4 pedine, la partita viene terminata con un pareggio. Il contatore dei pareggi viene incrementato, e viene mostrato all'utente il numero di partite vinte da ciascun giocatore oltre a quelle terminate in parità.Come nel caso in cui la partita è terminata con una vittoria, verrà chiesto all'utente se desidera giocare una nuova partita o se vuole uscire dal programma.

3. Composizione del programma

Il codice sorgente è composto dai seguenti file:

- ✓ forza4.c: parte principale del programma
- ✓ campogioco.h: file header contenente i prototipi delle funzioni e delle procedure implementate nel file campogioco.c
- ✓ campogioco.c: file contenente l'implementazione di 3 procedure che agiscono sul campo di gioco.
- ✓ stampe.h: file header contenente i prototipi delle procedure implementate nel file stampe.c
- ✓ stampe.c: file contenente l'implementazione di alcune procedure utilizzate per la stampa a video di messaggi per gli utenti.

3.1 Descrizione del file forza4.c

Il file forza4.c costituisce il cuore del programma, comprende la maggior parte delle funzioni e delle procedure utilizzate per la realizzazione del nostro programma inclusa ovviamente la la funzione "main".
Nella parte dichiarativa troviamo l'inclusione delle librerie standard "stdio.h" e "stdlib.h" oltre ai due file header che fanno parte del progetto sviluppato, "campogioco.h" e "stampe.h". Sono presenti inoltre la definizione del tipo dato strutturato "pl" che viene utilizzato per contenere le informazioni sui giocatori (ovvero il nome scelto, il simbolo che verrà visualizzato nel tabellone di gioco, ed il numero di vittorie che l'utente riuscirà ad ottenere), la definizione della matrice che sarà utilizzata come campo da gioco, costruita utilizzando le constanti RIGHE e COLONNE definite nel file header campogioco.h e i prototipi di 4 funzioni e due procedure:

- ✓ datigiocatori: Procedura utilizzata per richedere ai giocatori il proprio nome ed inserirlo all'interno della variabile strutturata pl grazie al puntatore che viene passato nel momento in cui viene richiamata;
- ✓ chiediposizione: funzione che chiede all'utente la colonna in cui inserire la propria pedina restituendo tale informazione al programma principale. Per il proprio funzionamento alla funzione vengono passati i puntamenti al campo da gioco e alla variabile strutturata pl;
- ✓ checkwin: funzione che verifica se nella matrice sono presenti 4 simboli uguali in orizzontale, verticale o diagonale. Al termine della verifica viene restituito un intero al programma principale. Per il funzionamento, la funzione ha bisogno di ricevere il puntamento al campo di gioco oltre a quello della variabile strutturata *pl*;
- ✓ fineturno: procedura che viene richiamata quando la partita viene vinta da uno dei due giocatori oppure quando termina con un pareggio. Si occupa di stampare a video il numero di vittorie ottenute dai giocatori dopo la partita appena terminata e di azzerare il contatore delle mosse utilizzato nel ciclo principale del programma per gestire il passaggio da un giocatore all'altro;

- ✓ rivincita: funzione che chiede ai giocatori se effettuare una nuova partita o uscire dal programma. Nel primo caso la matrice viene svuotata e la variabile "vittoria", a guardia del ciclo while principale viene reimpostata in modo da riprendere a ciclare fino alla fine di una nuova partita.

Nella parte esecutiva le prime operazioni riguardano l'inizializzazione delle variabili necessarie all'esecuzione del main. La prima procedura ad essere richiamata è *banneriniziale*, che stampa un messaggio di benvenuto all'utente, dopodiché vengono definite le due variabili strutturate di tipo *pl* (definita precedentemente). Mediante la chiamata alla procedura *datigiocatori* i giocatori inseriscono il proprio nome che verrà memorizzato dalla procedura stessa all'interno della relativa variabile strutturata, grazie al puntatore passato in fase di chiamata alla procedura stessa.
È presente un *getchar()* per permettere all'utente di rivedere le informazioni prima di proseguire.
La matrice viene inizializzata mediante la procedura *riempicampo*, dopodichè inizializzo le variabili intere *mosse* a 0 (avrà la funzione di contatore delle mosse e sarà utilizzata per gestire l'alternanza dei turni tra un giocatore e l'altro e per verificare se la matrice è piena) e la variabile *vittoria* a -1. Questa variabile sarà utilizzata come guardia dell'esecuzione del ciclo *while* richiamato successivamente, che verrà eseguito fino a che la variabile -1 resta impostata a questo valore. All'interno del ciclo *while* c'è un if concatenato con due *else if*, ed una ulteriore *if*. La prima if verifica se il tabellone è pieno, condizione che potrebbe verificarsi nel caso in cui nessun giocatore sia riuscito a mettere 4 pedine in linea. Nelle successive *else if* servono per far alternare i due giocatori. La prima verifica che l'operazione mosse mod 2 dia esito 0, la seconda che dia esito 1. Nel primo caso faccio giocare il giocatore 1, nel secondo il giocatore 2. Ovviamente le operazioni all'interno di questi due *else if* sono identiche, cambia solamente il riferimento alla variabile *pl* relativa al giocatore. Come prima cosa viene chiesto all'utente di scegliere la colonna dove inserire la pedina, tramite la funzione *chiediposizione,* alla quale viene passato il puntamento alla matrice e alla variabile del giocatore *pl.* Il puntamento alla matrice è necessario per verificare che l'utente abbia scelto una colonna che effettivamente esista e che non sia piena, mentre il puntamento alla variabile contenente i dati dei giocatori è necessario per leggere il simbolo assegnato al giocatore. La colonna selezionata viene memorizzata all'interno di una variabile che verrà passata, insieme al simbolo del giocatore (che questa volta viene passato come valore mediante una variabile di appoggio) alla procedura mettipedina, che si occuperà di inserire la pedina all'interno del campo da gioco. Una volta effettuata questa operazione, il contatore delle mosse viene incrementato e viene richiamata la funzione *checkwin*, richiamata passandogli i puntamenti alla matrice e alla variabile strutturata *pl*, che si occupa di verificare la presenza di 4 pedine allineate orizzontalmente, verticalmente, o sulle due diagonali. Il controllo viene fatto utilizzando delle *if* all'interno delle quali ci sono dei cicli *for* annidati che scorrendo la matrice verificano se effettivamente il giocatore sia riuscito a mettere in linea 4 pedine.
In caso di verifica positiva, viene stampato un messaggio a schermo che indica in quale modo le pedine si sono allineate (orizzontale, verticale, diagonale crescente o decrescente), e la variabile *vittoria* viene impostata ad un valore compreso tra 1 e 4 diverso per ogni casistica di allineamento rilevata. In realtà per questa versione del programma servirebbe semplicemente che il valore della variabile *vittoria* sia impostata ad un valore diverso da -1,

ma per lasciare spazio ad eventuali evoluzioni è stato scelto di differenziare le casistiche. Arrivati a questo punto viene verificata la condizione dell'ultima *if* , che controlla che la variabile vittoria sia diversa da -1. Questa condizione si verifica nel caso in cui ci sia un pareggio oppure la partita sia terminata pari. All'interno di questa selezione chiamo la funzione rivincita, che chiede all'utente se desidera rigiocare oppure se vuole uscire dal programma. Tale funzione riceve in input il puntamento alle variabili *count* e mosse, il puntamento alla matrice ed il valore della variabile partite. Il puntamento alla variabile *count* serve per tenere sotto controllo i tentativi errati nello scegliere se giocare e uscire e viene incrementata ad ogni scelta errata, quello alla variabile *mosse* serve per resettare il contatore mosse ed impostarlo a 0 nel caso in cui il valore ricevuto della variabile partite sia pari (viene passato solo il valore e non il puntamento perché è necessaria solamente la lettura di tale valore), mentre viene impostato a (1+(RIGHE*COLONNE)) nel caso in cui la variabile partite sia dispari. Questa differenziazione è stata effettuata per fare in modo che non sia sempre lo stesso giocatore ad iniziare la partita. Ovviamente la condizione dell'*if* all'interno del ciclo *while* principale che verifica se la matrice è piena verifica che il contatore *mosse* raggiunga il valore RIGHE*COLONNE (necessario quando il contatore count è inizialmente impostato a 0) oppure il valore (1+(2*RIGHE*COLONNE)), necessario quando il contatore è impostato a (1+(RIGHE+COLONNE)). Il puntamento alla matrice è necessario per reinizializzare il campo di gioco mediante la procedura *riempicampo* .

3.2 Descrizione del file campogioco.h

All'interno del file campogioco.h sono definite le costanti COLONNE e RIGHE che definiscono l'ampiezza del campo da gioco, la matrice composta da n righe e m colonne (i valori sono presi dalle costanti RIGHE e COLONNE), ed i i prototipi delle procedure che operano sulla matrice e che sono definite nel file campogioco.c:

- ✓ riempicampo;
- ✓ stampacampo;
- ✓ mettipedina.c .

Il funzionamento di tali procedure è spiegato nel paragrafo successivo.

3.3 Descrizione del file campogioco.c

Come anticipato nel paragrafo precedente, all'interno di questo file sono state raggruppate le definizioni delle procedure che operano sulla matrice che rappresenta il campo da gioco. La scelta è stata fatta per mantenere il programma ordinato e quindi effettuando le dovute separazioni.
Vediamo nel dettaglio il funzionamento delle procedure.

- ✓ riempicampo: questa procedura riceve in input solamente il puntamento alla matrice bidimensionale "campoGioco" (definito nell'header campogioco.h). Il compito che esegue è quello di scorrere la matrice mediante l'utilizzo di due cicli *for* annidati uno all'interno dell'altro, e ad ogni occorrenza inserisce il carattere "spazio".
- ✓ stampacampo: procedura che si occupa di stampare a video il contenuto della matrice. La matrice viene scorsa tramite due cicli *for* annidati, e ogni occorrenza

incontrata viene stampata a video tramite il comando *printf* insieme a dei caratteri separatori. Al di sopra e al di sotto della matrice vengono aggiunte delle linee di caratteri al fine di renderla più leggibile durante il gioco.

- ✓ mettipedina: questa procedura riceve in input la colonna scelta dall'utente e si occupa di inserirla nella prima posizione libera disponibile. La validazione dell'input inserito dall'utente viene fatto dalla funzione chiediposizione, quindi è sicuramente possibile inserire almeno una pedina nella colonna ricevuta come input. Oltre alla colonna viene passato il simbolo dell'utente che andrà inserito nella matrice (entrambe queste variabili sono passate per valore) e il puntamento al campo da gioco. Per effettuare le sue operazioni viene dichiarata una variabile di tipo intero *i* che verrà utilizzata per scorrere le righe al fine di trovare la prima posizione libera sulla colonna scelta dall'utente. Il codice è costruito con una *if* che verifica nella posizione più bassa (che corrisponde in realtà all'ultima riga, nel nostro caso "RIGHE - 1" non siano presenti altre pedine (in realtà viene verificata la presenza del carattere "spazio", inserito in fase di inizializzazione dalla procedura *riempicampo*). Se la condizione non è verificata, andiamo a ricadere in un *else* che scatena un ciclo *for* utilizzando la variabile *i* per scorrere la colonna alla ricerca della prima riga libera. Inizialmente la variabile viene posta al valore "RIGHE -1", ad ogni esecuzione del ciclo il contatore viene decrementato e l'esecuzione continua fino a che il contatore arriva a zero. Appena viene trovata una riga libera viene inserito il simbolo del giocatore e la variabile *i* viene posta a -1 in modo da uscire dal ciclo.

3.4 Descrizione del file stampe.h

All'interno del file stampe.h sono riportati esclusivamente i prototipi di due procedure, bannerіniziale e bannervittoria.

3.5 Descrizione del file stampe.c

Il file stampe.c contiene la definizione di due procedure che stampano a video dei banner realizzati in ascii-art tramite il comando *printf*. La scelta di racchiudere queste procedure in un file separato deriva dal fatto che hanno uno scopo simile e possono essere isolate dal resto del programma, non avendo bisogno di alcun input per essere richiamate. La prima, bannerіniziale, stampa a video un ascii-art riportante il nome del programma oltre a delle istruzioni. La seconda procedura, bannervittoria, stampa a video il messaggio "hai vinto!" all'utente.