SameGame

Progetto d'esame del corso di "Programmazione e Laboratorio di programmazione"

Appello del 9 settembre 2008

Scopo del progetto è realizzare un programma per giocare a SameGame (solitario noto anche coi vari altri nomi, tra cui Clickomania o Klickety, le cui regole sono descritte in seguito). Gli unici vincoli da rispettare sono i seguenti:

- il programma deve implementare le seguenti funzionalità:
  - leggere le mosse da standard input,
  - aggiornare il campo da gioco a seconda della mossa eseguita,
  - visualizzare il campo da gioco su standard output, con una qualunque modalità di visualizzazione basata sull'uso dei caratteri (non è richiesto e non verrà valutato l'utilizzo di particolari modalità di visualizzazione grafiche);
- il programma deve essere scritto in ANSI C (compilabile senza errori con gcc in laboratorio);
- non è consentito l'uso di librerie oltre alla *standard library*.

Per il resto, siete completamente liberi di definire quali caratteristiche debba avere il vostro videogioco e come svilupparlo.

Il codice sorgente deve essere accompagnato da una relazione sintetica (in formato `pdf` o `txt`) che illustri le caratteristiche del programma, le scelte fatte sia nella fase di definizione delle specifiche che nelle fasi di progettazione e implementazione, nonchè il modo in cui interfacciarsi con il programma.

La valutazione dipenderà da molti fattori, tra cui:

- scelta delle caratteristiche implementate (quantità e qualità),
- correttezza della loro implementazione,
- qualità dell'implementazione (principi di buona programmazione, tipi di dati usati, occupazione della memoria, ecc),
- stile del codice (indentazione, commenti, leggibilità, ecc),
- chiarezza, completezza e correttezza della relazione.

In questo documento vengono descritte le regole del gioco SameGame, forniti alcuni suggerimenti e definite la modalità di consegna degli elaborati. Siete liberi di optare per soluzioni diverse da quelle qui suggerite; in questo caso dovete motivare e documentare le scelte fatte nella relazione che descrive il progetto.

La discussione del progetto (alla quale è necessario presentarsi con una copia stampata della relazione e del codice) si svolgerà in data 11 settembre insieme alla prova orale. Orario e aula verranno specificati sul sito `http://lonati.dsi.unimi.it/progtelecom/`.

1 Regole del gioco

Le regole sono qui descritte in modo discorsivo, senza formalizzazione, e corredate da alcuni esempi illustrativi.

SameGame è un solitario. Il campo di gioco è costituito da un rettangolo inizialmente riempito con blocchi di dimensione unitaria e di colori diversi, disposti in modo casuale. L'obiettivo del gioco è rimuovere dal campo tutti i blocchi, selezionando via via delle aree di blocchi adiacenti dello stesso colore. Ad ogni mossa, il giocatore sceglie un blocco: se il blocco scelto è circondato (sopra, sotto, a destra, a sinistra) solo da blocchi di colori diversi, nessun blocco viene rimosso; in caso contrario, vengono rimossi tutti i blocchi nella più grande area di blocchi adiacenti a quello scelto e dello stesso colore. Ogni volta che vengono rimossi dei blocchi, il campo di gioco si *ricompatta* come segue: in ogni colonna, i buchi lasciati liberi dai blocchi rimossi vengono riempiti dai blocchi che si trovano sopra, i quali *cadono* verso il basso; le colonne eventualmente rimaste vuote vengono riempite dalle colonne alla loro destra, le quali *scivolano* verso sinistra.

1.1 Esempi

Consideriamo un campo di gioco di 10 colonne e 5 righe, numerate a partire da 0, da sinistra verso destra e dal basso verso l'alto, come in un sistema di coordinate cartesiane: ogni blocco è individuato da una coppia di posizioni (x,y), dove x denota la colonna e y la riga in cui si trova il blocco. Rappresentiamo i colori possibili con i simboli `| + o x -`.

All'inizio di una partita il campo di gioco potrebbe apparire come nella seguente figura (a sinistra). Il blocco cerchiato nella figura a destra, di colore +, è denotato dalla posizione $(1,3)$; selezionando tale posizione, non viene rimosso alcun blocco, poiché i blocchi ad esso adiacenti hanno tutti colore diverso da +.

4	-	|	+	-	-	+	|	-	+	+
3	|	+	|	o	-	+	o	x	+	x
2	+	o	+	+	x	|	x	x	x	|
1	|	-	|	+	+	|	-	-	x	x
0	x	-	-	-	o	-	x	x	x	+
	0	1	2	3	4	5	6	7	8	9

4	-	|	+	-	-	+	|	-	+	+
3	|	(+)	|	o	-	+	o	x	+	x
2	+	o	+	+	x	|	x	x	x	|
1	|	-	|	+	+	|	-	-	x	x
0	x	-	-	-	o	-	x	x	x	+
	0	1	2	3	4	5	6	7	8	9

Selezionando il blocco in posizione $(7,2)$, cerchiato nella seguente figura (a sinistra), verranno rimossi tutti i blocchi evidenziati in grassetto e il campo da gioco si ricompatterà come nella figura a destra.

4	-	|	+	-	-	+	|	-	+	+
3	|	+	|	o	-	+	o	**x**	+	x
2	+	o	+	+	x	|	**x**	(**x**)	**x**	|
1	|	-	|	+	+	|	-	-	**x**	**x**
0	x	-	-	-	o	-	**x**	**x**	**x**	+
	0	1	2	3	4	5	6	7	8	9

4	-	|	+	-	-	+				
3	|	+	|	o	-	+				+
2	+	o	+	+	x	|	|			x
1	|	-	|	+	+	|	o	-	+	|
0	x	-	-	-	o	-	-	-	+	+
	0	1	2	3	4	5	6	7	8	9

Ora, selezionando il blocco di posizione $(8,1)$, cerchiato nella seguente figura (a sinistra), verranno rimossi tutti i blocchi evidenziati in grassetto ed il campo da gioco si ricompatterà come nella figura a destra.

4	-	|	+	-	-	+				
3	|	+	|	o	-	+				+
2	+	o	+	+	x	|	|			x
1	|	-	|	+	+	|	o	-	⊕	|
0	x	-	-	-	o	-	-	-	**+**	**+**
	0	1	2	3	4	5	6	7	8	9

4	-	|	+	-	-	+				
3	|	+	|	o	-	+				
2	+	o	+	+	x	|	|		+	
1	|	-	|	+	+	|	o	-	x	
0	x	-	-	-	o	-	-	-	|	
	0	1	2	3	4	5	6	7	8	9

2 Suggerimenti

In questa sezione vengono dati alcuni suggerimenti sia per la fase di definizione delle caratteristiche del videogioco che per la fase di progettazione e di sviluppo. Siete liberi di optare per soluzioni diverse da quelle qui suggerite; in questo caso dovete motivare e documentare le scelte fatte nella relazione che descrive il progetto.

Definizione delle caratteristiche del videogioco

- I colori possono essere rappresentati con delle lettere oppure con altri simboli, come negli esempi precedenti.
- Si può prevedere la stampa del sistema di Coordinate, come negli esempi precedenti, in modo che il giocatore possa individuare le posizioni con più facilità.
- Si può prevedere che il giocatore scelga le dimensioni del campo di gioco e/o il numero di colori.
- Si possono prevedere dei messaggi d'errore nel caso in cui vengano scelte delle mosse non valide.
- Si può prevedere un criterio per dichiarare il gioco concluso (tale criterio va descritto nella relazione che accompagna il progetto), ad esempio: verifica dell'assenza di mosse valide, limite sul numero di mosse previste, limite sul numero delle mosse non valide consecutive, ecc.
- Si possono prevedere delle modalità DEMO, in cui il programma gioca da solo secondo certe strategie (tali strategie vanno descritte nella relazione che accompagna il progetto).
- Si può prevedere la possibilità di inserire un punteggio (le modalità di assegnazione dei punti devono essere descritte nella relazione che accompagna il progetto).

Progettazione e implementazione

- E' utile strutturare il programma usando delle funzioni, raggruppandole logicamente in file distinti. Es: funzioni di input/output (per leggere la prossima mossa, visualizzare il campo da gioco, ecc); funzioni di aggiornamento del campo (per calcolare l'area da rimuovere, far cadere i mattoni, far scivolare le colonne, ecc).
- E' utile definire dei tipi di variabili adatti a rappresentare gli elementi in gioco (blocchi, colonne, campo, posizioni, ecc).
- Ci sono molti modi per rappresentare il campo di gioco. Il modo più semplice prevede l'utilizzo di un array bidimensionale, ma naturalmente questo comporta delle limitazioni sulle sue dimensioni e un uso non ottimale della memoria: è buona cosa cercare altre soluzioni più efficienti.

- Per inizializzare il campo da gioco, potete usare le funzioni `time` (da `time.h`), `rand` e `srand` (da `stdlib.h`). La chiamata funzione `rand()` produce un numero apparentemente casuale, ma generato in realtà a partire da un seme. La funzione `srand(n)` inizializza il seme; se il seme non viene inizializzato, il suo valore di default è 1. La chiamata della funzione `time(NULL)` restituisce data e ora corrente, codificate come un unico intero; con la chiamata `srand(time(NULL))` è possibile differenziare i semi e quindi garantire che il campo da gioco venga inizializzato in modo diverso all'inizio di una nuova partita.

- Non è del tutto banale, dato un blocco, ottenere l'area da rimuovere. Si può procedere costruendo incrementalmente una sorta di *maschera*, definita come un array bidimensionale delle dimensioni del campo di gioco, le cui componenti possono assumere solo due valori: `on` in corrispondenza dei blocchi contenuti nell'area da rimuovere, `off` altrove. Inizialmente, solo la posizione corrispondente al blocco scelto è attiva (valore `on`); ad ogni passo, l'array viene scansito e, per ogni posizione attiva, vengono analizzate le posizioni adiacenti, le quali vengono attivate se e solo se corrispondono a blocchi del colore giusto nel campo di gioco. Questa soluzione non è particolarmente efficiente: ci sono algoritmi migliori per risolvere lo stesso problema, ad esempio basati sulla ricorsione, o sulle visite di grafo; chi fosse in grado di trovare soluzioni preferibili a quella descritta sopra è libero di implementarle.

- Per implementare una modalità DEMO si deve prevedere una strategia. Per definire una strategia, si può ad esempio fissare un criterio per la scelta della prossima mossa, ad esempio: scelgo sempre a caso; scelgo sempre il blocco rimovibile più a sinistra (e, a parità di colonna, il più in basso); scelgo sempre un blocco che permette di rimuovere il più grande numero possibile di blocchi; ecc.

- Si può prevedere un effetto di pseudo-animazione, in cui la visualizzazione del campo di gioco sovrascrive quella precedente. Potete usare l'istruzione `printf("\x1b[H")` per posizionare il cursore nell'angolo in alto a sinistra, e l'istruzione `printf("\x1b[H\x1b[2J")` per cancellare l'intero schermo. Nella modalità DEMO, l'aggiornamento del campo e la sua visualizzazione potrebbero risultare troppo rapidi: potete in tal caso usare l'istruzione `usleep(100000)`, da `time.h`, prima della stampa (ritarda l'esecuzione di un decimo di secondo).

3 Modalità di consegna

Il progetto deve essere svolto individualmente.

La consegna del progetto deve avvenire tramite l'invio di una sola mail, da spedire entro il giorno 7 settembre 2008 (incluso) all'indirizzo `lonati@dsi.unimi.it`. La mail deve tassativamente provenire dall'indirizzo di posta ufficiale dello studente, ossia l'indirizzo che termina con `@studenti.unimi.it`. La mail deve contenere un unico file allegato, di tipo archivio `zip`, il cui nome deve essere composto dal cognome seguito dal numero di matricola del studente (es: per lo studente Mario Rossi matr. 427481, il file deve chiamarsi `rossi427481.zip`); l'archivio deve contenere:

- il codice sorgente (compresi eventuali header file),
- la relazione in formato `pdf` o `txt`.

I sorgenti devono compilare tutti senza errori col compilatore gcc installato in laboratorio. **In presenza di errori di compilazione, la consegna non sarà ritenuto valida. Programmi consegnati diversamente da come specificato in questa sezione non saranno presi in considerazione.**

La discussione del progetto si svolgerà il giorno 11 settembre insieme alla prova orale (orario e aula verranno specificati sul sito `http://lonati.dsi.unimi.it/progtelecom/`). **E' necessario presentarsi alla prova orale con una copia stampata della relazione e del codice.**