Progetto Esame Programmazione Procedurale a.a. 2020/2021 - "Trannoi"

Canali per le domande: ricevimento, email: francesco.santini@unipg.it, Telegram: @safran

7 Dicembre 2020

Si realizzi un programma in linguaggio C che consista dei seguenti tre file (utilizzare "obbligatoriamente" questi tre nomi nel progetto):

- `main.c` contiene solo la definizione della funzione *main()*.
- `gamelib.c` contiene le definizioni delle funzioni che implementano il gioco.
- `gamelib.h` contiene le dichiarazioni delle funzioni definite in `gamelib.c` (solo quelle non *static*) e le definizioni del tipo delle strutture dati utilizzate in `gamelib.c`.

La storia. Un gruppo di astronauti si trova in viaggio sull'astronave Skelt, e il loro obiettivo è riuscire a completare tutte le attività previste (le *quest*) per il mantenimento della nave, arrivando così a destinazione. Tra di loro si celano però anche degli *impostori*, il cui scopo è eliminare di nascosto gli *astronauti* senza farsi scoprire da essi. Riusciranno ad arrivare a destinazione prima di essere decimati? Trannoi è liberamente ispirato ad un gioco esistente.[1]

main.c. Questo file contiene solo la funzione *main()*, il cui compito è stampare un menu di scelte verso il giocatore ed aspettare il suo comando. Il gioco viene giocato da 4 a 10 giocatori, suddivisi in astronauti ed impostori. Le possibili scelte sono: *1)* imposta gioco, *2)* gioca, *3)* termina gioco. *Suggerimento*: utilizzare un ciclo *do...while* per stampare il menu (dato che deve essere stampato per lo meno una volta), leggere la scelta del giocatore da tastiera, e con uno *switch* eseguire i comandi per effettuare la scelta richiesta. In caso il comando non sia 1-2-3, stampare un messaggio al giocatore che il comando è sbagliato, e poi ristampare il menu (stesso controllo su input per tutti gli altri menu stampati nel gioco). Nel caso la scelta sia 1, 2, o 3, chiamare la corrispondente funzione definita in *gamelib.c*, cioè *imposta_gioco()*, *gioca()*, e *termina_gioco()*. Non si può giocare se prima non si è impostato il gioco.

[1] `https://it.wikipedia.org/wiki/Among_Us`.

gamelib.h. Questo file deve contenere solamente le dichiarazioni delle funzioni definite in *gamelib.c* (come non *static*), e le definizioni dei tipi utilizzati nella libreria, cioè i tipi:

- *struct Giocatore*, contiene tre campi: la posizione della stanza in cui si trova il giocatore (un puntatore a *struct Stanza*), lo stato del giocatore (di tipo *enum Stato_giocatore*), il nome del giocatore (di tipo *enum Nome_giocatore*).
- *enum Stato_giocatore* deve rappresentare i valori {*astronauta, impostore, assassinato, defenestrato*}.
- *enum Nome_giocatore* deve rappresentare i valori {*rosso, blu, . . ., elencare 10 colori*}.
- *struct Stanza* contiene tre puntatori a *struct Stanza*, i cui identificatori sono *avanti*, *sinistra*, *destra*, ed un quarto puntatore (sempre a *struct Stanza*) il cui valore rappresenta il puntatore alla stanza precedente (*struct Stanza* stanza_precedente*). La stanza contiene anche un altro campo di tipo *enum Tipo_stanza*, che descrive cosa si trova in una data stanza. La stanza contiene anche un flag booleano *emergenza_chiamata* che indica se per questa stanza è stata effettuata una chiamata di emergenza.
- *enum Tipo_stanza* deve rappresentare i valori {*vuota, quest_semplice, quest_complicata, botola*}.

gamelib.c. Questo file rappresenta il nucleo del progetto e contiene tutte le altre funzioni necessarie:

Funzioni da implementare in gamelib.c. Questo file rappresenta il nucleo del progetto. Esso contiene le seguenti variabili globali: *struct Giocatore* giocatori* è un puntatore all'array di *struct Giocatore* che viene creato dinamicamente in base a quanti giocatori si vogliono nel gioco (4-10). La variabile *unsigned short quest_da_finire*, con il numero di *quest* da finire per vincere il gioco da parte degli astronauti. Un puntatore *struct Stanza* stanza_inizio* in cui viene memorizzata la prima stanza, dalla quale iniziano a giocare tutti i giocatori presenti. Un puntatore *struct Stanza* lista_stanze* che contiene il puntatore alla stanza di inizio, e che rappresenta la lista di tutte le *struct Stanza* create durante il gioco, una dopo l'altra in ordine di creazione. Questa lista serve come appoggio per scorrere tutte le stanze, per esempio per trovare quelle con le botole.

Le funzioni minime da implementare nel progetto sono:

a) La funzione *imposta_gioco()* viene richiamata da *main.c* e serve per richiedere i parametri per impostare il gioco; anche in questo caso, stampare un menu e leggere la risposta da tastiera. Questa funzione deve richiedere il numero di giocatori (da $n = 4$ a $n = 10$), e deve creare l'array di n giocatori. Per ogni giocatore scegliere a caso il nome (ogni giocatore si deve chiamare con un colore diverso), e decidere casualmente se *impostore* oppure *astronauta*. Ogni giocatore ha una probabilità di essere un impostore, e ci possono essere da 1 a 3 di loro per gioco (su n giocatori). All'aumentare dei giocatori (n) è più probabile che aumentino gli impostori. Implementare l'algoritmo per assegnare queste probabilità a scelta, seguendo questi principi. Ogni giocatore è rappresentato da una *struct Giocatore* memorizzata nell'array *struct Giocatore* giocatori*, che viene creato nello heap proprio in questa funzione. Questa funzione richiede anche il numero di *quest* da finire necessarie per vincere il gioco da parte degli astronauti: quando questo numero viene raggiunto, gli impostori hanno automaticamente perso. Questa funzione crea anche la prima stanza, assegnandola al puntatore *struct Stanza* stanza_inizio*. Tutti i giocatori iniziano il gioco in questa stanza iniziale, i cui i puntatori direzione ((*avanti*, *destra*, *sinistra*, *stanza_precedente*) sono inizializzati a *NULL*. Viene assegnato in modo casuale lo stato della stanza: 25% *botola*, 15% *quest_doppia*, 30% *quest_semplice*, 30% *vuota* (sommano a 100%).

Dalla *imposta_gioco()* possono essere chiamate le seguenti funzioni:

1) La funzione *stampa_giocatori* stampa tutte le informazioni relative ad ogni giocatore (memorizzate all'interno di ciascuna *struct Giocatore*).

2) La funzione *inizia_gioco()* esce dall'impostazione del gioco chiudendo questa fase: si può quindi giocare chiamando la funzione *gioca()* dal menu principale.

b) La funzione *gioca()* viene richiamata da *main.c* e da essa si possono chiamare tutte le azioni per giocare. I giocatori effettuano la loro mossa uno dopo l'altro, scegliendo ogni turno il giocatore in modo casuale. Il gioco è pensato a turni, prima è il turno del giocatore *nero*, poi *blu*, etc. Ad ogni turno un giocatore sceglie da un menu UNA sola possibile azione. All'inizio del proprio turno viene mostrato al giocatore il tipo della stanza, i giocatori presenti nella stessa stanza, e le possibili azioni (queste dipendono anche dal fatto che il giocatore sia *impostore* o *astronauta*). Le azioni sono rappresentate dalle funzioni *avanza()*, *esegui_quest()*, *chiamata_emergenza()*, *uccidi_astronauta()*, *usa_botola()*, *sabotaggio()*. Queste funzioni sono tutte definite come *static* in *gamelib.c*, dato che la loro visibilità non deve andare oltre a questo file. Ognuna delle seguenti funzioni/azioni "consuma" un turno di un giocatore.

1) La funzione *avanza()* fa scegliere al giocatore in quale direzione procedere rispetto alla stanza in cui si trova, oppure *rimanere fermo*. Può essere chiamata da *impostori* e *astronauti*. Se la direzione scelta (*avanti*, *destra*, *sinistra*) ha un puntatore *NULL* (cioè nessuno è ancora andato in quella direzione), viene creata una nuova stanza collegandola a questa attraverso il puntatore corrispondente (lista in memoria dinamica); notare che un giocatore non potrà mai tornare indietro: *struct Stanza* stanza_precedente* conterrà sempre il puntatore alla stanza a partire dalla quale è stata creata, ma un giocatore non potrà mai tornare in quella direzione. Il giocatore si sposta quindi in direzione *avanti*, *destra*, *sinistra*. In modo casuale viene assegnato lo stato della nuova stanza creata, con le stesse probabilità della prima stanza. Si deve ovviamente aggiornare anche la posizione del giocatore che si è spostato. La nuova stanza creata deve essere anche inserita in coda alla lista rappresentata dal puntatore globale *struct Stanza* lista_stanze* in modo da avere una lista di tutte le stanze (può servire per esempio per *usa_botola*).

2) La funzione *esegui_quest()* può essere chiamata da un *astronauta* se la stanza su cui si trova ha tipo *quest_semplice* o *quest_complicata*. Se semplice il contatore delle quest viene incrementato di uno, altrimenti di due. Una volta eseguita, la stanza diventa *vuota* e gli altri giocatori non possono finire la stessa *quest*. In caso si arrivi al limite *quest_da_finire*, il gioco corrente termina con la vittoria degli *astronauti*.

3) *L*a funzione *chiamata_emergenza()* può essere chiamata da tutti in caso la stanza in cui si trovi contenga un astronauta *assassinato*. In questo caso, c'è una probabilità che uno e uno solo tra tutti i presenti nella stanza venga *defenestrato* (un *defenestrato*, al pari di un *assassinato*, esce dal gioco). Per ogni *impostore* è del 30% più 20% per ogni altro *astronauta* nella stessa stanza, meno 30% per ogni altro *impostore* nella stessa stanza. Per ogni *astronauta* è del 30% più 20% per ogni altro *impostore* nella stessa stanza, meno 30% per ogni altro *astronauta* nella stessa stanza. Per ogni stanza può essere chiamata una ed una sola volta l'emergenza (utilizzare il flag *emergenza_chiamata* dell *struct Stanza*).

4) La funzione *uccidi_astronauta()* può essere chiamata solo da un *impostore*. L'impostore uccide uno degli *astronauti* presenti nella stessa stanza a scelta (che quindi cambia stato a *assassinato*), che quindi non potrà più muovere nei seguenti turni. Se nelle vicinanza ci sono altri astronauti, c'è una probabilità che venga scoperto e poi *defenestrato* dall'astronave, non potendo più muovere nei successivi turni. La probabilità è data dalla somma di 50% per ogni altro *astronauta* presente nella stessa stanza, più 20% per ogni astronauta che si trova nella stanza precedente (utilizzare il puntatore *struct Stanza* stanza_precedente*).

5) La funzione *usa_botola* può essere chiamata solo da un *impostore* se la stanza su cui si trova ha tipo *botola*. L'impostore cambierà posizione, muovendo in una delle stanze a caso con una botola, oppure, se non ce ne sono, in una stanza a caso.

6) La funzione *sabotaggio()* può essere chiamata solo da un *impostore* se la stanza su cui si trova ha tipo *quest_semplice* o *quest_complicata*. La quest viene cancellata: la stanza diventa *vuota*.

c) La funzione *termina_gioco* serve per terminare il gioco e uscire. Tutta la memoria allocata dinamicamente deve essere deallocata dinamicamente, così come quando si decide di impostare il gioco nuovamente con una nuova *imposta_gioco*: il percorso delle stanze deve essere deallocato, così come ogni altro eventuale spazio di memoria allocato in memoria dinamica.

Per compilare. Includere `gamelib.h` dentro `main.c` e dentro `gamelib.c` (i.e. *#include "gamelib.h"*). A questo punto si può compilare indipendentemente `main.c` e `gamelib.c`, con rispettivamente i comandi `gcc -c main.c` e `gcc -c gamelib.c` (vengono generati rispettivamente i file `main.o` e `gamelib.o`). Infine, per comporre i due file, linkarli con `gcc -o gioco main.o gamelib.o`. Aggiungere sempre i flag *-std=c11 -Wall* (per esempio `gcc -c game.c -std=c11 -Wall`), per essere sicuri di seguire lo standard *C 2011* e farsi segnalare tutti i *warning* rispettivamente. I *warning* vanno TUTTI rimossi.

Note finali. Il testo deve essere ritenuto una traccia da seguire il più possibile, ma lo studente è libero di modificare ed estendere alcune parti, con la possibilità di ottenere punti aggiuntivi nella valutazione nei casi in cui l'estensione sia giudicata considerevole. Si possono utilizzare solamente le funzioni di libreria standard del C[2]: contattare il docente in caso si vogliano utilizzare librerie differenti. Come ispirazione, possono essere utilizzati gli esempi svolti dei progetti degli anni passati presenti sulla pagina del corso[3]. Ricordarsi di aggiungere nome, cognome e matricola nel file README del progetto, su Github. Nel file README, oppure in un altro file di testo, descrivere brevemente le funzionalità in più in caso siano state aggiunte.

[2] `https://it.wikipedia.org/wiki/Libreria_standard_del_C`.
[3] `http://www.dmi.unipg.it/francesco.santini/progI.html`.

```c
#include <stdio.h>
#include <stdlib.h> // Da includere per utilizzare rand() e srand()
#include <time.h> // Da includere per utilizzare time()

int main () {
  time_t t;

  /* Inizializza il generatore di numeri casuali utilizzando il tempo attuale */
  srand((unsigned) time(&t));  // Funzione da chiamare una volta sola nel programma

  /* Ritorna un numero tra 0 e 99 */
  printf("%d\n", rand() % 100); // Chiamare quando si ha bisogno di un numero random
}

```

Figura 1: Esempio di come funziona la generazione di un numero casuale.