Progetto Esame Programmazione I

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

4 Dicembre 2017

Si realizzi un programma in linguaggio C che consiste dei seguenti tre file:

- `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. È venerdì 13 giugno 1980 ed un gruppo di studenti universitari del Corso di Laurea in Informatica si trasferiscono in vacanza al "Campeggio Lake Trasymeno", che sta per riaprire. Ventitre anni prima, infatti, nello stesso campeggio, un ragazzino di nome Gieson era annegato per colpa della negligenza di un programmatore: a causa di un *segmentation fault* nel suo programma di noleggio delle barche, alla famiglia di Gieson era stata affidata una barca con un motore difettoso. Gieson però infesta ancora il campeggio, e nutre vendetta nei confronti degli informatici poco attenti che hanno seguito il corso di Programmazione I giocando a League of Legends... Giacomo e Marzia, entrambi studenti del primo anno, rimangono isolati dagli altri quando...[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. Le possibili scelte sono: *1)* crea mappa, *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. Nel caso la scelta sia 1, 2, o 3, chiamare la corrispondente funzione definita in *gamelib.c*, cioè *crea_mappa()*, *gioca()*, e *termina_gioco()*. Una volta terminata la costruzione della mappa, si può costruirne un'altra (perdendo la prima: esiste una sola mappa giocabile), e non si può giocare senza prima aver costruito una mappa.

[1] Storia liberamente tratta da *Venerdì 13*: `https://en.wikipedia.org/wiki/Friday_the_13th_(1980_film)`.

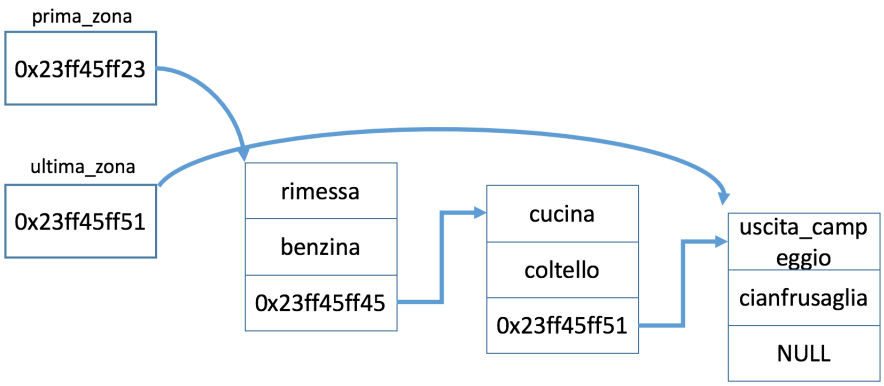

Figura 1: Esempio di lista dinamica con 3 zone.

gamelib.h. Questo file deve contenere solamente le dichiarazioni delle funzioni definite in *gamelib.c* come non statiche (vedere paragrafo successivo), e le definizioni dei tipi necessari alla libreria, cioè i tipi *struct Giocatore* e *struct Zona*, ed i tipi *enum Stato_giocatore*, *enum Tipo_zona* e *enum Tipo_oggetto*.

La *struct Giocatore* contiene *i)* un campo *enum Stato_giocatore stato*, *ii)* un campo che memorizza la posizione sulla mappa (*struct Zona* posizione*), e *iii)* un array *zaino* di 6 posizioni, ciascuna di tipo *unsigned short*: per ogni posizione si mantiene memorizzato il numero di oggetti di quel tipo trovati durante il gioco. A scelta, è possibile limitare il numero massimo di elementi dello zaino.

La *struct Zona* contiene i campi *i) enum Tipo_zona zona*, *ii) enum Tipo_oggetto oggetto*, *iii) struct Zona* zona_successiva*.

Il tipo *enum Tipo_zona* deve rappresentare i valori *cucina, soggiorno, rimessa, strada, lungo_lago, uscita_campeggio*.

Il tipo *enum Tipo_oggetto* deve rappresentare i valori *cianfrusaglia, bende, coltello, pistola, benzina, adrenalina*.

Il tipo *enum Stato_giocatore* deve rappresentare i valori *morto, ferito, vivo*.

gamelib.c. Questo file rappresenta il nucleo del progetto: esso contiene tre variabili globali, una *static struct Zona* prima_zona* (inizializzata a *NULL*) che punta alla prima zona della mappa, una *static struct Zona* ultima_zona* (inizializzata a *NULL*) che punta all'ultima zona della mappa, e due variabili *static struct Giocatore marzia* e *static struct Giocatore giacomo*. Le variabili devono essere *static* perché modificabili solamente tramite le funzioni della libreria (solo all'interno di *gamelib.c*). Alla partenza, lo zaino di Marzia contiene solamente due unità di adrenalina, quello di Giacomo un coltello (inizializzazione degli array *zaino* di *marzia* e *giacomo*: *zaino[adrenalina]= 2* e *zaino[coltello]= 1* rispettivamente).

La lista delle zone che rappresentano la mappa è un lista dinamica, dato che a priori non sappiamo il numero di zone nella mappa (è il giocatore a sceglierne il numero durante la creazione della mappa): il puntatore *prima_zona* punta alla prima zona, ed ogni zona che viene aggiunta attraverso la funzione *ins_zona()* (vedere dopo), richiamata dalla *crea_mappa()*, viene allocata in memoria dinamica utilizzando *malloc()*. Quando si deve eliminare dalla memoria dinamica, utilizzare *free()* sul puntatore della zona. All'interno di ogni zona, il campo *struct Zona* zona_successiva* punta alla zona successiva nella mappa (deve essere *NULL* in caso sia l'ultima zona della mappa). Alla fine di ogni inserzione (in *ins_zona()*), ricordarsi di salvare in *ultima_zona* il puntatore all'ultima zona inserita (altrimenti si deve scorrere dal'inizio la lista fino all'ultima zona quando si va ad inserire una nuova zona). Un esempio di lista dinamica con 3 zone è presentato in Figura 1.

Funzioni da implementare in gamelib.c. Le funzioni da implementare nel progetto sono:

a) La funzione *crea_mappa()* viene richiamata da *main.c* e serve per chiedere al giocatore le operazioni da effettuare sulla mappa; anche in questo caso, stampare un menu e leggere la risposta da tastiera. Esse corrispondono alle funzioni *ins_zona()*, *canc_zona()*, *stampa_mappa()*, e *chiudi_mappa()*. Tutte queste funzioni devono essere definite *static*, in quanto non devono essere chiamate da fuori *gamelib.c*

1) Inserimento di una zona in fondo alle terre già create (*ins_zona()*). Essa crea la nuova zona in memoria dinamica (*malloc()*), la inserisce nella lista modificando il valore del puntatore *zona_successiva* dell'ultima zona della lista con il risultato della *malloc()*, e aggiorna *ultima_zona* con il risultato della *malloc()*. Il tipo della zona (di tipo *enum Tipo_zona*) deve essere passato da tastiera. L'unico oggetto presente in quella zona (di tipo *enum Tipo_oggetto*) deve essere invece assegnato random con le probabilità dipendenti dal tipo della zona, come in Tabella 1. Una volta quindi passato il tipo della zona da tastiera, si può determinare anche randomicamente l'oggetto. Suggerimento: rappresentare le probabilità in Tabella 1 con una variabile matrice 6×6 (globale e *static*, con elementi *const*).
2) Cancella l'ultima zona inserita nella mappa (*canc_zona()*). Essa libera la memoria occupata dall'ultima zona ed aggiorna *ultima_zona* di conseguenza.
3) Stampa i campi di tutte le zone create fino a quel momento (*stampa_mappa()*), compreso il valore del puntatore alla zona successiva.
4) Fine della creazione della mappa *chiudi_mappa()*. Ci si ricorda che la mappa è stato terminata (per esempio settando una variabile globale e *static* da 1 a 0), e si esce anche dalla funzione *crea_mappa()*. Non si può chiudere la mappa se l'ultima zona inserita non ha tipo *uscita_campeggio* (questo è un controllo da effettuare). Non si può chiudere la mappa anche se ci sono meno di 8 zone inserite.

b) La funzione *gioca()* viene richiamata da *main.c* (punto 2). Marzia e Giacomo iniziano entrambi dalla prima zona della mappa. Sempre in modo random (probabilità 50% ciascuno), ogni turno è iniziato da uno dei due giocatori, e di seguito terminato dall'altro (a me no che un giocatore non usi *adrenalina* e compia azioni per più di un turno - vedi sotto -). Ad ogni turno un giocatore sceglie da un menu UNA sola possibile mossa. Le possibili mosse sono rappresentate dalle funzioni *avanza()*, *mostra_oggetto()*, *prendi_oggetto()*, *cura()*, e *usa_adrenalina()*. Queste funzioni sono tutte definite come *static* in *gamelib.c*.

1) Con la funzione *avanza()*, la posizione nel campo della struttura del personaggio in questione (*struct Zona* posizione*) viene aggiornato con quella della zona successiva nella mappa. In pratica, Marzia/-Giacomo avanzano di una zona verso l'uscita.
2) La funzione *mostra_oggetto()* serve per mostrare l'oggetto presente nella zona dove si trova in quel momento il personaggio. Questa funzione corrisponde a cercare nella zona corrente se esistono oggetti da poter poi prendere con la funzione seguente.
3) La funzione *prendi_oggetto()* serve per prendere l'oggetto presente nella zona dove si trova in quel momento il personaggio. Non si può prendere un oggetto se prima non è stato visualizzato con *mostra_oggetto()*: non si è a conoscenza della presenza o meno di un oggetto in una zona se prima non si è cercato con *mostra_oggetto()*. Se l'oggetto viene preso lo zaino viene aggiornato di conseguenza: per esempio, *zaino[coltello]++* se si trova un coltello. Aggiornare il campo *oggetto* della zona corrente a *cianfrusaglia*.
4) La funzione *cura()* permette di utilizzare una delle bende nel proprio zaino (se presente), per riportare il proprio stato di salute da *ferito* a *vivo*.
5) La funzione *usa_adrenalina()* serve per effettuare due mosse consecutive allo stesso giocatore. Quando questa mossa viene scelta, lo stesso giocatore può effettuare ulteriori due mosse, cioè due volte una delle funzioni *avanza()*, *mostra_oggetto()*, *prendi_oggetto()*, *cura()*, o *usa_adrenalina()*. L'adrenalina viene consumata di una unità (aggiornamento di *zaino*).

	cianfrusaglia	bende	coltello	pistola	benzina	adrenalina
cucina	30	20	40	0	0	10
soggiorno	20	10	10	30	0	30
rimessa	20	10	30	0	30	10
strada	80	0	10	0	10	0
lungo_lago	70	0	10	0	20	0
uscita_campeggio	90	0	10	0	0	0

Tabella 1: Probabilità di aggiungere in fase di creazione di una zona un dato oggetto ad una data zona.

Alla fine di ogni turno di Marzia e Giacomo, qualsiasi mossa abbiano effettuato, c'è la probabilità del 30% che appaia Gieson. La possibilità sale al 75% se ci si trova all'uscita del campeggio. In caso appaia, se non ci si può difendere si muore, se si possiede un coltello si rimane feriti (al secondo ferimento si muore), se si possiede una pistola non accade niente, se si possiede della benzina Gieson non apparirà nei prossimi 4 turni (cumulativi per Marzia e Giacomo). Il giocatore può scegliere come difendersi in caso ci siano più opzioni a disposizione, consumando ciò che ha utilizzato rimuovendolo dallo zaino.

Se Marzia o Giacomo muoiono o escono dalla mappa dopo un *avanza()* dalla zona *uscita_campeggio*, l'altro giocatore continua il gioco, ma la probabilità che Gieson appaia sale al 50% per ogni turno successivo.

c) La funzione *termina_gioco()*, dealloca tutto ciò che è stato memorizzato nella memoria *heap*, scorrendo la mappa ed utilizzando la funzione *free()* su ogni puntatore di ogni zona. Infine saluta il giocatore stampando un messaggio di "Game Over".

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 rimossi.

```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 100 */
  printf("%d\n", rand() % 101); // Chiamare quando si ha bisogno di un numero random
}

```

Figura 2: Esempio di come funziona la generazione di un numero casuale.

Possibili estensioni. Il testo deve essere ritenuto una traccia da seguire il più possibile, ma lo studente è libero di modificare ed estendere alcune parti. Una possibile estensione consiste nella lettura della mappa da file (ulteriori punti assegnati in fase di esame). Si possono utilizzare solamente le funzioni di libreria standard del C[2]: contattare il docente in caso si vogliano utilizzare librerie differenti.

[2] `https://it.wikipedia.org/wiki/Libreria_standard_del_C.`