Politecnico di Milano

Scuola di Ingegneria Industriale e dell’Informazione

**Insegnamento Fondamenti di Informatica (IMA - MEZ)**

**Anno Accademico 2021 / 2022**

Responsabile di Laboratorio: Edoardo GIACOMELLO [edoardo.giacomello@polimi.it](mailto:edoardo.giacomello@polimi.it)

Webpage Corso: <https://webeep.polimi.it/course/view.php?id=1374>

**Laboratorio #5**

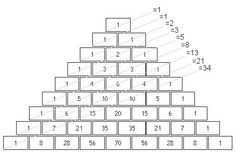
## File

**Problema 1**

Scrivere un programma che legge in input un numero intero positivo N e che costruisca la piramide di Fibonacci fino all’iterazione N.

Salvare la piramide su un file di nome fibonacci\_N.txt, con N il numero scelto.

Piramide di fibonacci:



Nota: un esempio di output per N=2 è:

1

11

121

**Suggerimento:** Per ottenere una stringa con il nome “fibonacci\_N.txt” dove N è una variabile intera, si suggerisce di utilizzare la funzione [sprintf](https://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm).

**Problema 2**

Scrivere un programma che legge un file testuale (lettura.txt) e legge da terminale un numero interno n positivo. Quindi salva in un file (output.txt) le prime n righe del file letto in precedenza.

**Problema 3**

Scaricare il file [album.bin](https://drive.google.com/file/d/0B4rpp0ph9e5hRFlwd2dCMXhBTzQ/view?usp=sharing&resourcekey=0-22A-Ejn3W9JGtW4RLTqt_g) che contiene una sequenza di elementi del seguente tipo:

typedef struct

{

char artista[50];

char titolo[50];

int num; //numero traccia

int minuti;

int secondi;

} traccia;

Scrivere un programma che:

* legga il contenuto di album.bin
* stampi le informazioni di tutte le tracce del CD
* consenta all’utente di modificare la durata di una traccia scelta da lui
* salvi la modifica all’interno del file

**Progetto finale**

Provare a sviluppare un prototipo di videogioco di tipo [Roguelike](https://it.wikipedia.org/wiki/Roguelike).

**Step1**

Il nostro videogioco si svolgerà all’interno di una mappa costituita da una griglia quadrata, che rappresenta un labirinto da cui il giocatore deve cercare di uscire.

Ciascun quadrato della griglia può ospitare diversi elementi del gioco: un muro, il giocatore, l’uscita del labirinto o uno spazio vuoto calpestabile.

Per semplicità possiamo rappresentare questa griglia con una matrice di interi, usando i seguenti valori per codificare ciascun elemento della griglia a seconda del tipo:

* spazio vuoto → 0
* muro → 1
* giocatore → 2
* uscita labirinto → 3

Possiamo inoltre salvare una mappa in un file di testo così organizzato: la prima riga conterrà la dimensione della griglia (un singolo valore intero, dato che la griglia è quadrata) e nelle righe successive il file conterrà la matrice che definisce la mappa di gioco (memorizzata nel file riga per riga). Un esempio di file che contiene una mappa di gioco utilizzando questa struttura è disponibile [qui](https://drive.google.com/file/d/0B4rpp0ph9e5hVGprUW41Vi1FMUE/view?usp=sharing&resourcekey=0-U61Pk57tw5gQGY4ZaFXr1g).

Implementare un semplice programma che legge il file di esempio e visualizza su schermo la griglia di gioco, usando un carattere diverso per ciascun elemento della mappa.

Ad esempio, utilizzando i seguenti caratteri:

* spazio vuoto → ‘ ‘
* muro → ‘#’
* giocatore → ‘K’
* uscita labirinto → ‘O’

la mappa di esempio dovrebbe apparire a video come:

#######

#K#####

# ##

### ###

## O###

#######

#######

**Suggerimenti**

1. Dal momento che la matrice ha dimensioni variabili, si suggerisce di utilizzare un doppio puntatore per memorizzarla, ad esempio il seguente frammento di codice alloca lo spazio di memoria sufficiente a contenere nrighe x ncol interi:  
     
   int \*\*map,i;  
   map = calloc(nrighe,sizeof(int\*));  
   for (i=0; i<nrighe; i++)

map[i] = calloc(ncol,sizeof(int));

1. Per caricare la mappa da file, si suggerisce di implementare una funzione specifica:  
     
   int \*\*load\_map(int \*size, const char filename[]);  
     
   che riceve in ingresso l’indirizzo di una variabile size che ospiterà la dimensione della mappa letta da file, il nome del file che contiene la mappa (filename); la funzione restituirà il contenuto della mappa caricata da file attraverso un doppio puntatore intero (come suggerito al punto precedente).
2. Per visualizzare la mappa, si suggerisce di implementare una funzione specifica:  
     
   void visualize\_map(int \*\*map, int size);  
     
   che riceve in ingresso la mappa da visualizzare (map) e la sua dimensione (size).

**Step2**

Una caratteristica distintiva dei giochi Roguelike è quella di offrire al giocatore delle mappe di gioco generate casualmente. Come secondo passo perciò proverete ad implementare il seguente algoritmo ricorsivo per la generazione di un labirinto:

**Input:** Una griglia bidimensionale N x M, in cui tutte le celle contengono un muro (valore 1).

**Output**: Una griglia bidimensionale N x M in cui è stato 'scavato' un labirinto, totalmente connesso e senza cicli, di pavimenti, circondato da muri.

**Algoritmo.** L'algoritmo ha come obiettivo la generazione di un labirinto, partendo da un punto all'interno della griglia bidimensionale (per semplicità possiamo ipotizzare di partire sempre dalla cella 1,1). Per far ciò, l'algoritmo si muoverà tra le caselle con indici (x,y) dispari (cioè spostandosi di due celle per volta), 'scavando' di volta in volta dei passaggi nei muri.

Ad ogni passo l’algoritmo riceve la mappa, la dimensione della mappa, la *cella corrente* (due indici x e y) e una struttura dati di supporto delle stesse dimensioni della mappa che utilizza per tenere traccia delle celle già visitate.

Per prima cosa, l’algoritmo marca come visitata la cella corrente ed inoltre la “scava” (cioè le assegna il valore 0 corrispondente ad uno spazio vuoto). Quindi procede come segue:

1. finché esistono celle dispari adiacenti (vedi **Nota1**) non già visitate:
2. si sceglie casualmente una delle celle dispari adiacenti
3. si “scava” un passaggio verso la cella scelta (cioè si assegna il valore 0 alla cella scelta e a quella che si trova fra la cella corrente e la cella scelta).
4. si richiama quindi l’algoritmo facendo diventare la cella scelta la nuova *cella corrente.*

Al termine dell’algoritmo, è possibile completare il labirinto definendo la posizione iniziale del giocatore e la posizione finale (l’uscita) del labirinto scegliendo una cella a caso fra quelle contenenti uno spazio vuoto.

**Nota1**: per cella dispari adiacente si intende non immediatamente adiacente, ma saltando di 2 indici alla volta in modo da ottenere indici dispari. Per esempio, la cella (5,5) avrà come adiacenti le celle (7,5), (5,7), (3,5), (5,3).

**Nota2**: l'algoritmo 'scava' le celle con indici x ed y dispari perché in questo modo ci sarà sempre un muro tra due passaggi paralleli, permettendo di ottenere un labirinto. Affinché il labirinto sia correttamente circondato da un muro, è necessario che la dimensione del labirinto sia dispari.

Utilizzando l’algoritmo appena descritto, implementare una funzione:

int \*\*generate\_map(int size);

che genera (utilizzando l’algoritmo ricorsivo appena descritto) e restituisce una mappa di dimensioni size.

Implementare anche la funzione:

void save\_map(int \*\*map, int size, const char filename[]);

che salva la mappa in ingresso (map), di dimensione size, in un file testo di nome filename, seguendo la struttura già descritta nello step precedente del progetto.

Quindi implementare un programma che consente ad un utente di scegliere la dimensione della mappa da generare e il nome del file in cui salvarla.

**Step3**

Implementare quindi un programma che consente di caricare una mappa da un file e di esplorarla alla ricerca dell’uscita.

Il giocatore partirà dalla posizione iniziale ed avrà a disposizione 4 azioni di movimento (alto, basso, destra e sinistra): ciascuna azione risulterà in uno spostamento di una cella nella direzione scelta dal giocatore se e soltanto se la cella di destinazione non contiene un muro.

Il gioco termina quando la casella di destinazione contiene l’uscita del labirinto oppure quando l’utente decide di abbandonare il gioco.

Provando il programma implementato con [questa mappa](https://drive.google.com/file/d/0B4rpp0ph9e5hYmlBem9hbnZ0bjQ/view?usp=sharing&resourcekey=0-poUbxest3DQ7WEwe2y1w-w), si dovrebbe ottenere questo risultato:

#####################

#K # # #

##### # ### ### # # #

# # # # # # #

# ##### # ### ### ###

# # # # # #

##### ### ##### ### #

# # # # # # #

# # ### # # # # # # #

# # # # # # # #

# ### ####### ##### #

# # # # # #

# ##### ### # # # ###

# # # # # # #

# # ##### ##### ### #

# # # O #

##### ##### ####### #

# # # # #

# ####### ##### # ###

# # #

#####################

Time: 0

Choose your next action (wasd - q to exit):

**Suggerimenti**

Per consentire al giocatore di muoversi nelle quattro direzioni fondamentali, si suggerisce di leggere un carattere utilizzando lo schema ‘w’ ‘a’ ‘s’ ‘d’ per decidere la direzione di movimento (‘w’ è su, ‘a’ è sinistra, ‘s’ è giù e ‘d’ e destra). Si può quindi utilizzare un altro carattere (ad esempio ‘q’) per abbandonare il gioco.

**Step4**

Modificare il programma scritto al passo precedente in modo che in ogni istante di gioco venga visualizzata soltanto una porzione della mappa di gioco.

In particolare, sia W la dimensione della finestra che si intende visualizzare (si suggerisce di provare con W=20 e utilizzare [questa mappa](https://drive.google.com/file/d/0B4rpp0ph9e5hVU41UDktcmk3Zzg/view?usp=sharing&resourcekey=0-SvMnpCBzOMA_s3WmYe18ag) per provare il programma), verrà visualizzata solo la porzione di mappa che si estende di W/2 celle a destra, a sinistra, in basso e in alto a partire dalla posizione attuale del giocatore (che sarà quindi sempre al centro).

**Nota:** Nel caso la porzione di mappa così definita vada al di fuori dei bordi della mappa (a sinistra, a destra, in basso o in alto) si rappresentino le celle mancanti come se fossero celle vuote.

**Suggerimenti**

Si suggerisce di modificare la funzione visualize\_map nel modo seguente:  
  
void visualize\_map(int\*\* map, int size, int px, int py, int win\_size);

dove i nuovi parametri sono la posizione del player (px e py) e la dimensione della finestra (win\_size); questa nuova funzione non visualizzerà l’intera mappa ma soltanto una porzione in cui il player al centro e si estende per win\_size/2 celle in ogni direzione.

**Passi successivi**

Estendere il gioco con nuovi elementi come:

* consentire il salvataggio di una partita
* rendere casuale la posizione di uscita del labirinto e la posizione iniziale del giocatore
* aggiungere alcune statistiche del giocatore (es. numero di turni di gioco, vita, punti esperienza, fame, etc.)