Lettura con `fgets()` e `sscanf()`

Per gli scopi del corso di Fondamenti di Informatica, è necessaria la lettura di file strutturati (di testo) nei casi dove il numero di righe contenute è esplicitamente dichiarato e quello dove non lo è.

L'esempio seguente tratta il primo caso: il file di nome `campionato1.txt` (riportato alla fine del documento) contiene i nomi e le squadre di appartenenza di `20` piloti.
Per prima cosa bisogna allocare lo spazio necessario per contenere i contenuti del file, ad esempio un vettore; inoltre dobbiamo ispirarci alla struttura di una riga per dichiarare, appunto, una struct adeguata; in questo caso le proprietà utili di una riga sono il nome del pilota e della sua squadra (righe `10-11`). Per comodità abbiamo definito un nuovo tipo a partire dalla struct `classificato`, chiamato `"t_class"`(ificato) (riga `14`) come nel documento #6-7.

Allochiamo quindi il vettore di tipo `t_class` di dimensione `20` (NB perché sappiamo esplicitamente il numero di righe) e successivamente apriamo il file, e chiamiamo la funzione `caricaDati()` (riga `44`) con argomenti il puntatore a `FILE` del file ed il vettore dei partecipanti appena creato. Notiamo che il passaggio dell'array avviene per riferimento al suo puntatore: nella funzione `caridaDati()` ci aspettiamo un puntatore a `partecipanti[]` (appunto `int *partecipanti[]`), in questo modo tutto ciò che scriviamo dentro questo vettore all'interno della funzione `caricaDati()` rimarrà nel vettore anche se dopo usciamo dalla funzione, quindi saremo in grado di utilizzarlo nel main. Se avessimo dichiarato il vettore dentro la funzione `caricaDati()` e avessimo restituito il puntatore a questo vettore al main, una volta usciti dalla stessa, tutti i dati a quell'indirizzo (i.e. il puntatore) sarebbero persi! Questo perché il vettore è trattato come una variabile locale (privata) di `caricaDati()`, la quale, una volta terminata, sparisce con tutte le variabili ivi dichiarate.

Sempre all'interno di `caricaDati()` utilizziamo la già vista `fgets()`, stavolta combinata in un ciclo `while`, dove esso verrà eseguito finche non si raggiunge la fine del file (`EOF`), in tal caso la `fgets()` ci restituirà un puntatore `NULL`, uscendo così dal ciclo.
Nel ciclo introduciamo la nuova funzione `sscanf(sorgente,"formattazione", destinazioni)` la quale si occuperà di prelevare il contenuto della "riga" sorgente (in questo caso `buff`), formattarlo secondo il layout voluto e sistemare le sotto-stringhe e/o i valori dentro le variabili di destinazione (i.e. la struct `classificato` all'i-esimo posto del vettore `partecipanti[i]`). La `sscanf()` è praticamente uguale alla `scanf()` vista precedentemente, a parte la sorgente, nel qual caso caso era `stdin`.
Ricordiamo che restiamo in un ciclo fino alla fine del file, bisogna quindi ricordarsi di aumentare l'indice `i` per andare avanti nell'array `partecipanti[i]`. Un lettore attento, noterà che invece di valutare la condizione `!= NULL` si può semplicemente valutare `i<20`, oppure addirittura un ciclo `for` fino a `20`. Il primo caso rimane comunque più generale, ma la scelta è data al lettore.
Una volta caricati i dati, dal main siamo in grado di operare sul vettore `partecipanti[]`, ad esempio stampiamo a video i contenuti del file secondo la formattazione che preferiamo composta nella `printf()`.

```c
//### Lettura file semplice, ovvero "sapendo il numero di righe" ###

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define NUM_CLASS 20

struct classificato {
  char pilota[25];
  char squadra[25];
};
//creo un nuovo tipo che chiamo "t_class"(ificato) a partire da "struct classificato"
typedef struct classificato t_class;

//NB qui ci sono delle novità rispetto lez. precedente
void caricaDati(FILE *fin, t_class *partecipanti){

  char buff[50];
  int i = 0;

  while( fgets(buff, sizeof(buff), fin) != NULL ){

    sscanf(buff,"%s %s", partecipanti[i].pilota, partecipanti[i].squadra);

    i++;
  }
}

int main() {

  //ptr al file di tipo FILE
  FILE *campionato;
  int i;

  //lo spazio concreto dove metterò i dati caricati dal file
  //NB è fisso, SO ESPLICITAMENTE IL NUMERO DEI CLASSIFICATI (me lo dice il testo...)
  t_class partecipanti[NUM_CLASS];

  //apro il file con fopen(nome_file, "in lettura")
  campionato = fopen("campionato1.txt","r");

  //passo i due puntatori
  caricaDati(campionato, partecipanti);

  for(i=0;i<NUM_CLASS;i++)
    printf("# %d: %s - %s\n", i+1, partecipanti[i].pilota , partecipanti[i].squadra);

  //chiusura file
  fclose(campionato);

  return 0;
}
```

lett_file_avanzato.c - Sorgente esempio

```
Valtteri_Bottas Williams
Carlos_Sainz Toro_Rosso
Max_Verstappen Toro_Rosso
Felipe_Nasr Sauber
Fernando_Alonso McLaren
Daniel_Ricciardo Red_Bull
Daniil_Kvyat Red_Bull
Pastor_Maldonado Lotus
Felipe_Massa Williams
Romain_Grosjean Lotus
Sebastian_Vettel Ferrari
Nico_Rosberg Mercedes
Roberto_Merhi Marussia
Marcus_Ericsson Sauber
Lewis_Hamilton Mercedes
Will_Stevens Marussia
Sergio_Perez Force_India
Nico_Hulkenberg Force_India
Kimi_Raikkonen Ferrari
Jenson_Button McLaren
```

campionato1.txt - Lista piloti F1 (fonte: Tutorato UNIPV del 10/12/2017)