# Programmazione I (Tucci/Distasi)

PR1 INT MT/RD 20/12/2022

Modello: 1

Cognome: _____

Nome: _____

Matricola: _____

Email: _____

**Regole del gioco:** Compilare i dati personali prima d'incominciare. Una volta iniziata la prova, non è consentito lasciare l'aula. Usare questi stessi fogli (compreso il retro, dove necessario) per rispondere. *Buon lavoro!*

1. Consideriamo la seguente struttura dati.

   ```
   typedef struct vendita
   {
     char desc[50];
     double prezzo;
     double quant;
     char venditore[15];
   } Vendita;
   ```

   Scrivere una funzione

   ```
   Vendita * load_vendite (FILE *fp, int *n)
   ```

   che legge un file binario e crea un array di oggetti di tipo `Vendita` allocato dinamicamente. Il file in questione contiene come primo dato un `int` che indica il numero di oggetti presenti nel file, poi i dati veri e propri. La funzione restituisce l'array così ottenuto, e usa il parametro output `n` per comunicarne la dimensione.

   In caso di problemi di qualsiasi tipo (per esempio: file di formato errato o fallita allocazione memoria), la funzione restituisce `NULL`, stampando su `stderr` un messaggio di errore che specifica l'errore incontrato.

2. Scrivere una funzione

```
Vendita * elenco_venditore (FILE *fin, char *id_venditore, int *nv)
```

che usa la funzione scritta al punto precedente per leggere un file binario e restituisce un array con tutte e sole le vendite relative al venditore indicato come parametro, usando il parametro output `nv` per comunicarne la dimensione. L'array restituito deve essere dimensionato esattamente per la misura necessaria. Quale potrebbe essere un valore ragionevole da restituire se il venditore indicato non ha vendite in archivio?

Saranno preferite le soluzioni che usano una funzione apposita per confrontare oggetti `Vendita`, tenendo presente che normalmente le `struct` si passano per riferimento.

# Risposte per il modello 1

1. Consideriamo la seguente struttura dati.

```
typedef struct vendita
{
  char desc[50];
  double prezzo;
  double quant;
  char venditore[15];
} Vendita;
```

Scrivere una funzione

```
Vendita * load_vendite (FILE *fp, int *n)
```

che legge un file binario e crea un array di oggetti di tipo `Vendita` allocato dinamicamente. Il file in questione contiene come primo dato un `int` che indica il numero di oggetti presenti nel file, poi i dati veri e propri. La funzione restituisce l'array così ottenuto, e usa il parametro output `n` per comunicarne la dimensione.

In caso di problemi di qualsiasi tipo (per esempio: file di formato errato o fallita allocazione memoria), la funzione restituisce `NULL`, stampando su `stderr` un messaggio di errore che specifica l'errore incontrato.

**Risposta** Ecco una possibile soluzione.

```
Vendita *load_vendite(FILE * fp, int *n)
{
  int n_in_file, status;
  Vendita *vendite = NULL;
  char *thisfunct = "load_vendite";     // for error messages

  status = fread(&n_in_file, sizeof(int), 1, fp);       // how many?
  if (status != 1)
    {
      simple_error(thisfunct, "can't read size from file");
      return NULL;
    }
  if (n_in_file == 0)
    {
      *n = n_in_file;           // set output parameter
      simple_error(thisfunct, "warning: input file says size==0");
      return NULL;
    }
  vendite = malloc(sizeof(Vendita) * n_in_file);
  if (vendite == NULL)
    {
      simple_error(thisfunct, "malloc() failed");
      return NULL;
    }
  status = fread(vendite, sizeof(Vendita), n_in_file, fp);
  if (status != n_in_file)
    {
      simple_error(thisfunct, "not enough data in file");
      return NULL;
    }
  *n = n_in_file;              // else, everything is OK
  return vendite;
}
```

```
void simple_error(char *function_name, char *err_msg)
{
  fprintf(stderr, "error: %s in function %s()\n", err_msg, function_name);
}
```

2. Scrivere una funzione

   ```
   Vendita * elenco_venditore (FILE *fin, char *id_venditore, int *nv)
   ```

   che usa la funzione scritta al punto precedente per leggere un file binario e restituisce un array con tutte e sole le vendite relative al venditore indicato come parametro, usando il parametro output `nv` per comunicarne la dimensione. L'array restituito deve essere dimensionato esattamente per la misura necessaria. Quale potrebbe essere un valore ragionevole da restituire se il venditore indicato non ha vendite in archivio?

   Saranno preferite le soluzioni che usano una funzione apposita per confrontare oggetti `Vendita`, tenendo presente che normalmente le `struct` si passano per riferimento.

   **Risposta** Ecco una possibile soluzione.

```
#include "vendita.h"

// questa vendita e' del venditore id_venditore?
int match_venditore(char *id_venditore, Vendita * v)
{
  return (strcmp(id_venditore, v->venditore) == 0);
}

Vendita *elenco_venditore(FILE * fin, char *id_venditore, int *nv)
{
  Vendita *vendite_tutte, *vendite_filtrate;
  int i, j, n, matches;


  vendite_tutte = load_vendite(fin, &n);        // all sales in here
  matches = 0;                  // count sales for this seller
  for (i = 0; i < n; i++)
    {
      if (match_venditore(id_venditore, &vendite_tutte[i]))
        {
          matches++;
        }
    }
  *nv = matches;
  if (matches == 0)
    {
      return NULL;              // but this is not an error
    }
  vendite_filtrate = malloc(matches * sizeof(Vendita));
  if (vendite_filtrate == NULL)
    {
      simple_error("elenco_venditore", "malloc() failed");      // this is an error
      exit(-1);
    }
  for (i = 0, j = 0; i < n; i++)
    {
      if (match_venditore(id_venditore, &vendite_tutte[i]))
        {
          vendite_filtrate[j++] = vendite_tutte[i];
        }
    }
  free(vendite_tutte);          // Don't waste memory that we don't need anymore
  return vendite_filtrate;
}
```