

# Elenco di esercizi C+Unix

Enrico Bini

October 16, 2022

## Premessa

Segue un elenco di esercizi assegnati durante l'anno.

**Importante** Per massimizzare l'apprendimento, si raccomanda di leggere la soluzione soltanto **dopo** aver provato a risolvere l'esercizio autonomamente. Difatti, leggere una soluzione scritta da altri dopo aver provato a risolvere il problema da soli permette di capire meglio la soluzione proposta.

## Esercizi C

**Es. [es-array-cat-err]** Il codice di `es-array-cat-err.c` dovrebbe concatenare due stringhe. Si corregga il codice affinché vengano effettivamente concatenate le due stringhe `v1` e `v2`. Ai fini di questo esercizio **NON** si possono utilizzare le funzioni di libreria `strcat`, `strncat`, `strlen`, etc. L'obiettivo è confrontarsi con errori tipici (sia di compilazione che run-time) che si possono ottenere, **NON** scrivere il codice di una funzionalità di cui esistono già molte implementazioni.

**Es. [es-array-odd-even]** Si legga da standard input (con `fgets+strtol`) un array di 7 interi. Si stampino prima tutti gli elementi di indice dispari e poi quelli di indice pari. Se per esempio vengono letti: 11 20 37 45 51 69 75, allora viene stampato: 11 37 51 75 20 45 69.

**Es. [es-sum-next]** Si legga da standard input (con `fgets+strtol`) un array `v1` di 10 interi. Si costruisca un altro array `v2` in cui:

- il primo elemento è la somma di tutti gli elementi di `v1`
- il secondo elemento è la somma degli elementi di `v1` a partire dal secondo
- ...
- l'ultimo elemento è uguale all'ultimo elemento di `v2`.

**Es. [count-char]** Si scriva un programma che legga una stringa da `stdin` e, per ogni carattere presente nella stringa, scriva a `stdout` una riga con il numero di occorrenze del carattere nella stringa e il carattere stesso. Per esempio, se la stringa letta è

Ciao a tutti!!

venga stampato a `stdout`

```
2,  
2,a  
2,i  
1,o  
3,t  
1,C  
1,u  
2,!
```

in ordine a piacere.

Si gestisca il caso in cui la stringa ecceda i limiti.

**Es. [print-after]** Si scriva un programma che legge due stringhe di caratteri (**s1** e **s2**) di lunghezza massima di 80 caratteri mediante **fgets**.

Preliminarmente, elimina i caratteri non stampabili da entrambe le stringhe **s1** e **s2** scrivendo il byte 0 sul primo byte non stampabile (un byte è stampabile se ha codice ASCII compreso fra 32 e 126). Ricorda: **fgets** memorizza nella stringa anche il carattere “a capo” che deve quindi essere eliminato.

- Se **s2** è contenuta all'interno di **s1**, il programma stampa la parte di **s1** che segue **s2**.
- Se **s2** non è contenuta all'interno di **s1**, non stampa niente.

Per esempio, se le stringhe **s1** e **s2** sono rispettivamente:

```
Ciao a tutti  
ia
```

allora verrà stampato

```
o a tutti
```

Si realizzi tale programma:

1. evitando di includere le funzioni della libreria **string.h**
2. evitando le parentesi quadre per riferire gli elementi di **s1** e **s2**

**Es. [get-exponent]** Si scriva un programma che:

1. legga un **double** da tastiera,
2. estraiga l'esponente della sua rappresentazione in floating point secondo lo Standard IEEE 754-1985
3. stampi tale esponente in decimale.

Suggerimento: si provi a leggere la memoria dove il numero floating point è memorizzato, come un intero **unsigned long** da cui poi estrarre l'esponente attraverso la manipolazione dei suoi bit.

**Es. [binary]** Si scriva un programma che legge un intero senza segno da tastiera **stdin** e scrive sul terminale la sua rappresentazione in base 2. Si eviti di usare gli operatori di divisione **/** e di resto **%**, preferendo invece gli operatori bitwise e quelli di shift. Si eviti di usare **strtoul(s, NULL, 2)** che fa esattamente questo.

**Es. [triangle-star]** Scrivere un programma che stampi a video un triangolo rettangolo di **'\*** la cui base e altezza siano lette da tastiera. Esempio, se vengono inseriti 10 (base) e 4 (altezza), viene stampato quanto segue

```
*  
****  
*****  
*****
```

**Es. [caotic-seq]** Si consideri la successione generata dal numero **n** e che calcola il numero successivo come segue:

- se **n** è pari allora il prossimo numero è la metà di **n**
- se **n** è dispari allora il prossimo numero è il triplo più uno.

La sequenza termina quando si raggiunge 1.

Si scriva un programma che, accettato un valore numerico intero **N** da tastiera, stampi la lunghezza di tutte le sequenze generate per ciascun valore di partenza da 1 a **N**.

Es. [exam-2019.01.28] Implementare la funzione con prototipo

```
int range_of_even(int * nums, int length, int *min, int *max);
```

La funzione ha quattro parametri:

- `nums` è un array di numeri interi;
- `length` è la dimensione di `nums`;
- `min` e `max` sono puntatori usati dalla funzione per restituire degli interi al chiamante.

La funzione deve determinare il valore massimo e minimo **dei valori pari** presenti in `nums`. Se tali limiti esistono allora la funzione deve restituirli tramite i puntatori `min` e `max` al chiamante e restituire 1. Se l'array non contiene alcun numero pari, la funzione deve restituire 0 e i valori in `*min` e `*max` non saranno significativi.

Es. [fibo] Si realizzi la funzione con prototipo

```
int * fibo(int n);
```

la quale alloca e restituisce un array di `n` interi contenente i primi `n` numeri della successione di Fibonacci ([https://it.wikipedia.org/wiki/Successione\\_di\\_Fibonacci](https://it.wikipedia.org/wiki/Successione_di_Fibonacci)).

Inoltre si scriva la funzione `main` che legge `n` da tastiera, stampa gli elementi di `fibo(n)` e infine dealloca l'array.

Es. [sort-record] Data la seguente struct

```
typedef struct {
    char * name;
    int age;
} record;
```

si scriva il corpo delle seguenti funzioni:

```
record * rec_rand_create(int n);
void rec_sort(record * v, int n);
void rec_print(record * v, int n);
void rec_free(record * v, int n);
```

- La funzione `rec_rand_create` alloca e restituisce un array di `n` elementi di tipo `record` in cui
  - ogni stringa `name` contiene caratteri casuali e ha lunghezza casuale fra 1 e `MAX_LEN` (costante del pre-processore opportunamente definita)
  - ogni campo `age` è casuale fra `MIN_AGE` e `MAX_AGE`

Si veda `man 3 rand` per la generazione di numeri casuali

- la funzione `rec_sort` ordina gli elementi dell'array `v` di lunghezza `n` secondo il campo `age` crescente
- la funzione `rec_print` stampa l'array
- la funzione `rec_free` dealloca la struttura dati

Si realizzi quindi un `main` che testi le tre funzioni.

**Es. [list]** Si estenda il file `test-list.c` aggiungendo le seguenti funzioni:

1. la funzione

```
list list_insert_ordered(list p, int val);
```

che riceve in input una lista ordinata per valori crescenti puntata da `p` e inserisce il nuovo elemento `val` nella lista mantenendo l'ordinamento;

2. la funzione

```
list list_cat(list before, list after);
```

che riceve in input due liste `before` e `after` e restituisce in uscita la lista `before` a cui è stata aggiunta in coda la lista `after`

3. la funzione

```
list list_insert_tail(list p, int val);
```

che inserisce l'elemento `val` in coda alla lista puntata da `p` e ritorna la lista modificata

**Es. [list-more]** A partire dal file `es-list.c` realizzato nell'esercizio `[list]`, si realizzino anche le seguenti funzioni (ispirate a esercizi dello scritto):

1. la funzione

```
list list_delete_if(list head, int to_delete);
```

la quale cancella e dealloca il primo nodo della lista il cui valore del campo `value` è uguale al parametro `to_delete`. La funzione restituisce la lista così modificata.

2. la funzione

```
list list_delete_odd(list head);
```

la quale rimuova dalla lista ogni elemento in posizione dispari (il primo, il terzo, etc.). La funzione restituisce la lista così modificata.

3. la funzione

```
list list_cut_below(list head, int cut_value);
```

la quale rimuova dalla lista ogni elemento che abbia valore inferiore al valore `cut_value` passato come parametro.

4. la funzione

```
list list_dup(list head);
```

la quale ritorna una copia dalla lista (copia di ogni elemento).