

# Esercitazione 3

## Argomenti: funzioni e passaggio dei parametri

Scaricare il file [es3\\_funzioni.c](#). Aggiungere in [es3\\_funzioni.c](#) la definizione delle funzioni indicate negli esercizi seguenti. Modificare opportunamente la funzione `main` per effettuare delle verifiche di funzionamento delle funzioni scritte.

### Esercizio 3.1

Scrivere la funzione C

```
int* allocInt();
```

che alloca dinamicamente in memoria un intero e ne restituisce in uscita il puntatore.

### Esercizio 3.2

Scrivere la funzione C

```
void printInt(int *i1, int *i2);
```

che, dati in ingresso due puntatore ad intero `i1` e `i2`, ne stampi a schermo il valore

### Esercizio 3.3

Scrivere la funzione C

```
int MCD(int i1, int i2);
```

che, dati in ingresso due interi `i1` e `i2`, ne calcoli, e stampi a schermo, il massimo comun divisore

### Esercizio 3.4

Scrivere la funzione C

```
int mcm(int i1, int i2);
```

che, dati in ingresso due interi `i1` e `i2`, ne calcoli, e stampi a schermo, il minimo comune multiplo.

## Esercizio 3.5

Scrivere la funzione C

```
void conversioneTemperatura(int t, char c);
```

che, dati in ingresso un intero `t` e un char `c` rappresentanti rispettivamente un valore di temperatura e la scala di temperatura scelta ed effettui la conversione nelle altre scale e ne stampi a schermo il risultato. Le scale da considerare sono Celsius (carattere "C"), Kelvin (carattere "K") e Fahrenheit (carattere "F"). Formule di conversione:

$$K = C + 273.15$$

$$F = C * 9/5 + 32$$

## Esercizio 3.6

Scrivere la funzione C

```
void* conversioneTemperatura(int *t, char c);
```

che, dati in ingresso un puntatore ad intero `t` e un char `c`, svolga lo stesso compito della funzione precedente. Le due conversioni risultanti (ovvero due valori di temperatura, e due caratteri rappresentati la scala) devono essere salvate in una zona di memoria allocata dinamicamente con un'unica chiamata alla funzione `malloc()` e restituite in uscita.

## Esercizio 3.7

Scrivere la funzione C

```
void printConversione(void *temperatura);
```

che, dato in ingresso un puntatore a `void`, stampi il risultato della funzione precedente.

## Esercizio 3.8

Scrivere la funzione C

```
void soluzioneSistemaLineare(int i1, int i2);
```

che, dati in ingresso due interi `i1` e `i2`, risolva l'equazione:

$$i1 \times i2 = 0$$

e ne stampi il risultato a schermo.

## Esercizio 3.9

Scrivere la funzione C

```
void differenzaPuntatori(int *i1, int *i2);
```

che, dati in ingresso due puntatori ad intero `i1` ed `i2`, ne calcoli la distanza in memoria (tramite differenza di puntatori) e, successivamente, modifichi il contenuto di `i2` scrivendo una espressione che contiene solo il puntatore a `i1` e la differenza in memoria tra `i1` e `i2`.

## Esercizio 3.10

Scrivere la funzione C

```
void fibonacci(int N);
```

che, dato in ingresso un intero `N`, calcoli e stampi i primi `N` numeri della serie di Fibonacci. La serie di Fibonacci inizia con 1, 1 ed ogni numero successivo è dato dalla somma dei due precedenti: 1, 1, 2, 3, 5, 8, 13, 21, . . .

## Altri Esercizi Proposti

Creare un programma che sulla base di una selezione fatta attraverso l'input di un carattere consenta all'utente di acquisire i parametri di input ed invocare le funzioni sui numeri interi definite in precedenza (3.10, 3.4, 3.3) e stampi i rispettivi risultati. L'input e l'output dei risultati dovranno essere realizzati da opportune funzioni.

## Argomento: Array

Scaricare il file [es3\\_array.c](#). Aggiungere in [es3\\_array.c](#) la definizione delle funzioni indicate negli esercizi seguenti. Modificare opportunamente la funzione `main` per effettuare delle verifiche di funzionamento delle funzioni scritte. Per testare le funzioni bisognerà utilizzare degli array generati randomicamente.

## Esercizio 3.11

Scrivere la funzione C

```
void vec_print(double v[], int dim);
```

che, dato in ingresso un vettore `v`, di dimensione `dim`, stampi il vettore nel seguente formato:

```
[x_1 x_2 ... x_i ... x_dim]
```

## Esercizio 3.12

Allocare due vettori

```
v2 = [2.1, -3.5, 1.0, 6.5, -5.2]
v3 = [4.8, 0.1, -6.2, -2.5, 7.2]
```

Allocando il primo in maniera statica ed il secondo in maniera dinamica. Calcolare inoltre la dimensione del vettore `v2` utilizzando la funzione `sizeof()`

## Esercizio 3.13

Scrivere la funzione C

```
double* vec_sum(double v[ ], int dim);
```

che, dato in ingresso un vettore `v` di dimensioni `dim`, allochi e restituisca un vettore `v` dove ogni elemento è la somma di tutti i valori a lui successivi.

Esempio:

```
v = [ 1.0 -0.1 3.4 7.2 -5.3];
output = [ 6.2 5.2 5.3  1.9 -5.3 ]
```

## Esercizio 3.14

Scrivere la funzione C

```
double* vec_rec(double v1[ ], double v2[ ], int dim1, int dim2);
```

che, dato in ingresso due vettori `v1` e `v2` e le loro rispettive dimensioni `dim1` e `dim2`, allochi e restituisca un vettore `v` di dimensioni `dim1` dove l'i-esimo elemento sarà quello del vettore `v1` se ne esiste una ricorrenza nel vettore `v2`, altrimenti 0.

```
v1 = [ 1.0 -0.1 3.4 7.2 -5.3];
v2 = [ 1.0 2.5 7.2 ];
output = [ 1.0 0 0 7.2 0 ];
```

