Esercitazione 4 - 19/10

Argomenti:

- "cosa stampa?", tracciamento delle variabili e ricerca degli errori
- funzioni: senza parametri, con parametri, con `return`

Indice degli esercizi

Esercizio	Argomento
Esercizio 1	Numeri casuali: cosa restituisce
Esercizio 2	Numeri casuali: costruire l'intervallo
Esercizio 3	Numeri casuali: il gioco del dado
Esercizio 4	Numeri casuali e massimo
Esercizio 5	Do-while: cosa succede durante l'esecuzione
Esercizio 6	Do-while: scrivere la condizione
Esercizio 7	Condizioni e variabili: cosa stampa
Esercizio 8	Funzioni senza return
Esercizio 9	Funzioni: return
Esercizio 10	Funzioni e variabili locali
Esercizio 11	Funzioni e controllo dell'input: potenza
Esercizio 12	Funzioni e accumulatori: somma dei pari
Esercizio 13	Funzioni, numeri casuali e do-while
Esercizio 14	Indovina il numero

Esercizio 1

Per ciascuna espressione dite: se i valori sono **interi o decimali**, qual è il **minimo** e il **massimo** possibile (e se il massimo può uscire davvero) e **quanti valori diversi** possono uscire, se sono in numero finito.

A. `Math.random() * 10`

B. `Math.floor(Math.random() * 10)`

C. `Math.floor(Math.random() * 10) + 1`

D. `Math.floor(Math.random() * 6) + 1`

E. `Math.floor(Math.random() * 10) * 2`

F. `Math.floor(Math.random() * 3) - 1`

G. `Math.floor(Math.random() * 10 + 1)`

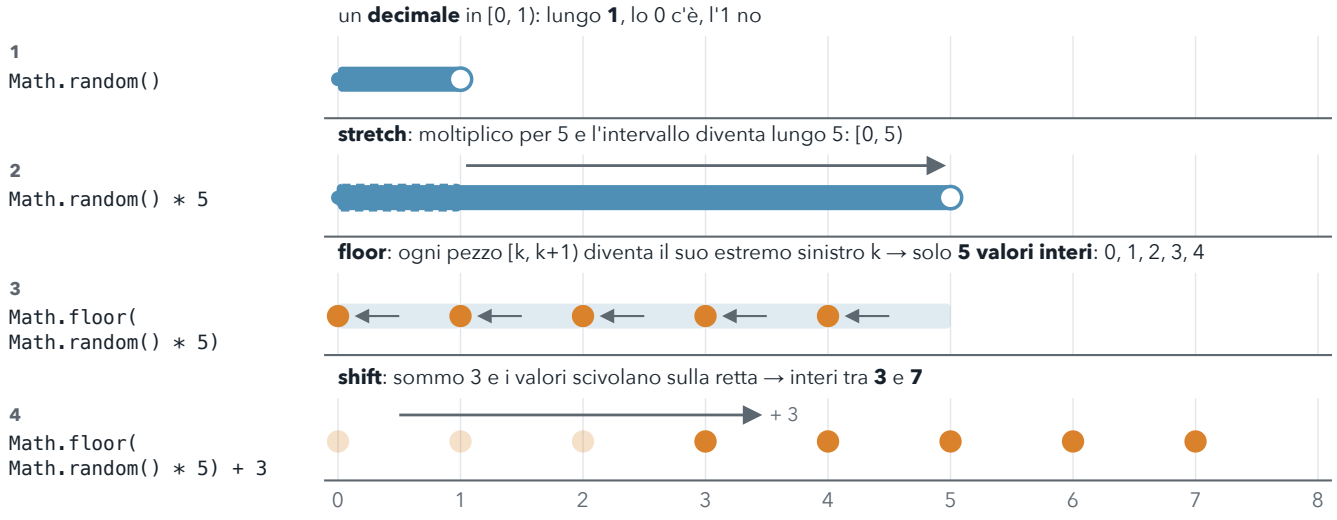
H. `Math.floor(Math.random() * 51) + 50`

Poi rispondete:

- se volete un intero tra `50` e `100`, perché si moltiplica per `51` e non per `50`?

Da `Math.random()` a un intervallo

Un intero casuale in un intervallo si costruisce in due mosse: prima si **allunga** l'intervallo di `Math.random()` (che è lungo 1), poi lo si **sposta** sulla retta. Esempio: un intero tra `3` e `7`.



`Math.floor(Math.random() * (max - min + 1)) + min`

stretch di (max − min + 1) valori, poi shift di min

Esercizio 2

Usate lo stesso schema della figura precedente negli intervalli seguenti. Completate (al posto di `_____`) i due numeri, in modo che l'espressione dia un **intero** nell'intervallo indicato (estremi inclusi):

```
Math.floor(Math.random() * _____) + _____
```

A. tra `0` e `3` (il secondo numero sarà `0`)

B. tra `1` e `20`

C. tra `3` e `7`

D. tra `10` e `20`

E. tra `-5` e `5`

Esercizio 3

Il gioco del dado. L'utente sceglie un numero tra 1 e 6, e il programma lancia un dado finché non esce **proprio quel numero**, contando quanti **tentativi** servono. A passi:

A. Simulare **un lancio** di dado: salvare il risultato in una variabile `lancio` e stampare `È uscito: ...`.

B. Il numero da ottenere è fisso: il `6`. Ripetere i lanci **finché non esce un 6**, stampando ogni lancio e, alla fine, quanti tentativi sono serviti. Completate (al posto di `_____`):

```
let tentativi = _____;
let lancio;
_____ {
  lancio = Math.floor(Math.random() * _____) + _____;
  tentativi++;
  console.log("Tentativo", tentativi + ":", lancio);
} _____ (_____);
console.log("Ci sono voluti", tentativi, "tentativi");
```

C. Ora il numero lo sceglie l'utente: chiedere un **intero tra 1 e 6** (se non lo è, ripetere la domanda) e poi lanciare il dado finché non esce quel numero. Stampare i tentativi alla fine.

Esempio di esecuzione (i lanci cambiano a ogni esecuzione), con `4` come numero scelto:

```
Tentativo 1: 2
Tentativo 2: 6
Tentativo 3: 4
Ci sono voluti 3 tentativi
```

Esercizio 4

Generare **10 numeri casuali interi tra 50 e 100** (estremi inclusi), stamparli e, alla fine, stampare il **più grande**. A passi:

A. Per trovare il più grande si usa un accumulatore, `massimo`. Compilate la tabella per i numeri `7, 3, 9, 4` che arrivano nell'ordine:

```
let massimo = 0;
// per ogni numero:
if (numero > massimo) {
  massimo = numero;
}
```

Numero	numero > massimo	massimo dopo
7		
3		
9		
4		

B. Scrivete il programma completo: 10 numeri casuali tra 50 e 100, stampati uno per riga, e alla fine il più grande.

Esercizio 5

Questo programma chiede un numero positivo finché l'utente non ne inserisce uno:

```
let x;
do {
  x = Number(prompt("Numero positivo:"));
  console.log("letto", x);
} while (x <= 0);
console.log("fine", x);
```

A. L'utente inserisce, nell'ordine, la sequenza `0`, `-2`, `5`. Cosa succede durante l'esecuzione? Per ogni giro dite quale valore viene letto, se il ciclo **continua o si ferma** e perché, e cosa viene **stampato**.

B. Stessa domanda con la sequenza `-7`, `0`, `1.5`, `4`. Tutti i numeri della sequenza vengono letti?

C. Con quale sequenza di input il ciclo fa **un solo giro**?

Esercizio 6

Per ciascuna richiesta, scrivete la **condizione del `while`**: deve essere vera quando il valore `n` è **sbagliato**, e quindi si richiede l'input.

A. Un numero qualsiasi (anche decimale) **maggiore di 0**. Completate: `while (_____)`

B. Un numero (anche decimale) tra `0` e `1`, estremi **inclusi**. Completate: `while (_____ || _____)`

C. Un numero **intero**, qualsiasi. Completate: `while (_____)`

D. Un numero **intero** tra `1` e `12` (estremi inclusi).

E. Un numero **intero** tra `1` e `12` (estremi **esclusi**).

Poi per ciascuna controllate la condizione con questi valori: `0`, `1`, `2.5`, `12`, `-3` (quali vengono accettati?).

Per casa. Stessa richiesta (scrivete la condizione del `while`) per queste altre situazioni. Controllate ogni condizione con qualche valore valido e qualche valore sbagliato, specialmente agli estremi.

F. Un numero **pari** (quindi intero). Suggerimento: `%` dà il resto della divisione. Provate con `0`, `12`, `7`, `2.5`, `-10`.

G. Un numero **intero positivo di 3 cifre** (da `100` a `999`). Provate con `99`, `100`, `999`, `1000`, `150.5`.

H. Un numero **negativo oppure maggiore di 10**. Attenzione: qui i valori accettati sono due intervalli separati. Provate con `-5`, `0`, `10`, `10.5`, `12`.

I. Un numero **intero tra 1 e 100** (estremi inclusi) che **non sia un multiplo di 10**. Provate con `0`, `7`, `10`, `55`, `100`.

J. Un numero (anche decimale) compreso tra `-10` e `10` (estremi inclusi) ma **non compreso tra `-1` e `1`** (estremi esclusi, cioè `-1` e `1` sono accettati). Provate con `-10`, `-1`, `-0.5`, `0`, `1`, `5`, `10`, `10.5`.

Esercizio 7

Per ogni programma, dite cosa stampa **senza eseguirlo**.

A.

```
let x = 7;
if (x > 5 && x < 10) {
  console.log("dentro");
} else {
  console.log("fuori");
}
```

B. Riguardate il programma A: cosa stampa se `x` vale `10`? E se vale `5`?

C.

```
let x = 7;
if (x < 5 || x > 10) {
  console.log("fuori");
} else {
  console.log("dentro");
}
```

D. Che cosa stampano `y` e `z`?

```
let x = 3;
let y = x > 2 && x < 5;
let z = !y || x == 3;
console.log(y, z);
```

E. Quali giri del ciclo fanno `n = n + i`? Cosa stampa?

```
let n = 0;
for (let i = 1; i <= 5; i++) {
  if (i % 2 == 0 || i == 5) {
    n = n + i;
  }
}
console.log(n);
```

F. Compilate la tabella con il valore di `a` e `b` dopo ogni istruzione. Cosa stampa?

```
let a = 4;
let b = 9;
a = a + b;
b = a - b;
a = a - b;
console.log(a, b);
```

Passo	Istruzione	a	b
1	`let a = 4;`		
2	`let b = 9;`		
3	`a = a + b;`		
4	`b = a - b;`		
5	`a = a - b;`		

Recap 1 · Funzioni

Una **funzione** è un blocco di istruzioni con un nome, che si può **richiamare** quante volte si vuole. Si definisce una volta con `function`; si usa scrivendo il nome con le parentesi.

```
function messaggio() {                          // senza parametri
  console.log("Benvenuti!");
}

messaggio();                                    // stampa: Benvenuti!
messaggio();                                    // stampa: Benvenuti!
```

Attenzione: la sola definizione **non esegue** niente; il codice nella funzione parte quando la si **richiama**.

I **parametri** sono i valori che la funzione riceve; quelli passati alla chiamata sono gli **argomenti**:

```
function saluta(nome) {                         // con un parametro
  console.log("Ciao " + nome + "!");
}

saluta("Giulia");                               // stampa: Ciao Giulia!
saluta("Marco");                                // stampa: Ciao Marco!
```

Le variabili dichiarate in una funzione (compresi i parametri) esistono **solo dentro** la funzione.

Per far **restituire** un valore si usa `return`:

```
function quadrato(x) {
  return x * x;
}

let q = quadrato(5);                          // q vale 25
console.log(quadrato(3) + 1);                 // stampa 10
```

- `return` **termina** la funzione e consegna il valore a chi l'ha chiamata, che può salvarlo o usarlo in un'espressione;
- `console.log` invece **stampa** e basta: non restituisce niente (il risultato di una funzione senza `return` è `undefined`);
- le istruzioni scritte dopo un `return` non vengono eseguite.

Due tipi di funzione, quindi: quelle che **fanno qualcosa** (come stampare), e quelle che **calcolano un valore** e lo restituiscono.

Esercizio 8

A. Cosa stampa questo programma? E se si **toglie** l'ultima riga?

```
function messaggio() {
  console.log("Benvenuti!");
}
console.log("inizio");
messaggio();
console.log("fine");
messaggio();
```

B. Scrivere la funzione `ripeti(testo, n)`, che stampa `testo` per `n` volte (una per riga). Ad esempio `ripeti("ciao", 3)` stampa `ciao` tre volte. Poi chiamatela con altri valori, ad esempio `ripeti("ok", 5)`.

Esercizio 9

A. Cosa stampa? Qual è la differenza tra le due funzioni?

```
function raddoppia(x) {
  return x * 2;
}
function stampaDoppio(x) {
  console.log(x * 2);
}
raddoppia(4);
stampaDoppio(4);
```

B. Cosa stampa? Attenzione a quello che succede nella variabile `b`.

```
function stampaDoppio(x) {
  console.log(x * 2);
}
let b = stampaDoppio(4);
console.log(b);
```

C. Il parametro è una copia. Cosa stampano le due `console.log`?

```
function aumenta(n) {
  n = n + 1;
  return n;
}
let n = 5;
aumenta(n);
console.log(n);
let m = aumenta(n);
console.log(m, n);
```

D. Cosa stampa? Quale `return` viene eseguito in ciascuna chiamata?

```
function segno(x) {
  if (x > 0) {
    return "positivo";
  }
  if (x < 0) {
    return "negativo";
  }
  return "zero";
}
console.log(segno(-3), segno(0), segno(8));
```

Esercizio 10

Scrivere le seguenti funzioni, che **restituiscono** un valore (non lo stampano). Provarle con qualche chiamata, stampando il risultato con `console.log`.

A. `maggiore(a, b)` restituisce il maggiore tra due numeri. Ad esempio `maggiore(3, 8)` vale `8`.

B. `perimetro(base, altezza)` restituisce il perimetro di un rettangolo; poi usatela per stampare il perimetro di un rettangolo `4 x 6`.

Nelle prossime serve una **variabile locale**: una variabile dichiarata **dentro** la funzione, in cui salvare un risultato intermedio.

C. `media_aritmetica(a, b, c, d)` restituisce la media aritmetica di quattro numeri: prima calcolate la loro somma in una variabile locale `somma`, poi restituite la media. Provatela con `31, 27, 18, 25`. Poi provate a stampare `somma` **fuori** dalla funzione: cosa succede?

D. `sconto(prezzo, percentuale)` restituisce il prezzo scontato. Calcolate in una variabile locale `risparmio` quanto si risparmia, e restituite `prezzo - risparmio`. Ad esempio `sconto(80, 25)` vale `60`.

E. Una festa ha `p` partecipanti e `b` bottiglie di birra, ciascuna da `c` cl. Un bicchiere contiene 100 ml (cioè 10 cl). Scrivete la funzione `festa(p, b, c)` che restituisce **quanti bicchieri** può bere in media ogni partecipante. Usate variabili locali per i passaggi intermedi (ad esempio i bicchieri totali). Provatela con `festa(10, 3, 75)` (risultato `2.25`), `festa(50, 15, 50)` e `festa(3, 2, 75)`.

Esercizio 11

La funzione `potenza(base, esponente)` restituisce `base` elevato a `esponente`, **senza usare** `**`: si moltiplica `base` per se stesso `esponente` volte. Funziona solo se `esponente` è un numero **intero non negativo** (anche `0`).

Scrivere la funzione `potenza`, con una variabile locale `risultato` che accumula i prodotti. Poi scrivere il programma che chiede all'utente la `base` e l'`esponente`: **prima di passare l'esponente alla funzione**, controllare che sia un **intero non negativo**, e se non lo è ripetere la domanda. Alla fine stampare il risultato.

Provatelo con base `2` ed esponente `10` (risultato `1024`), poi con esponente `0` (risultato `1`) e con esponenti non validi: `-3`, `2.5`, `abc`.

Suggerimento: per accumulare un **prodotto**, da quale valore deve partire `risultato`?

Esercizio 12

Scrivere la funzione `sommaPari(n)` che **restituisce** la somma dei numeri **pari** compresi tra `1` e `n` (estremi inclusi), cioè `2 + 4 + 6 + ...` fino a `n`. Ad esempio `sommaPari(10)` restituisce `30` (`2 + 4 + 6 + 8 + 10`) e `sommaPari(7)` restituisce `12`.

Poi, nel programma principale: chiedere `n` all'utente e stampare `sommaPari(n)`; stampare inoltre `sommaPari(10) - sommaPari(5)` e dire cosa rappresenta.

Esercizio 13

Scrivere due funzioni:

- `casuale(min, max)`, che **restituisce** un intero casuale tra `min` e `max` (estremi inclusi);
- `leggiIntero(min, max)`, che chiede all'utente un **intero tra `min` e `max`** (estremi inclusi), ripete la domanda finché il valore non è valido e **restituisce** il valore valido.

Poi usarle nel programma principale: chiedere il numero di un mese con `leggiIntero(1, 12)` e stampare un lancio di dado con `casuale(1, 6)`. Provare anche `casuale(50, 100)` e `casuale(-5, 5)`.

Esercizio 14

Indovina il numero. Il programma sceglie un numero segreto **casuale tra 1 e 100** (senza stamparlo!). L'utente prova a indovinarlo: a ogni tentativo il programma risponde `Troppo basso` oppure `Troppo alto`, finché l'utente non indovina. Alla fine il programma stampa in **quanti tentativi** ci è riuscito. Ogni tentativo deve essere un **intero tra 1 e 100**.

Suggerimento: usate le funzioni `casuale` e `leggiIntero` dell'esercizio 13.

 Ludovica Pannitto - Università di Pisa - CC BY-NC-SA 4.0