Esercitazione 2 - 05/10

Argomenti:

- tipo `boolean` e operatori di confronto
- condizioni: `if`, `else if`, `else`
- operatori logici `&&`, `||`, `!`
- `&&` e `||` con valori non booleani (truthy e falsy)
- ciclo `for`: somma e prodotto dei primi n numeri
- ciclo `while`: come alternativa al `for` e per condizioni diverse dal contare

Indice degli esercizi

Esercizio	Argomento
Esercizio 1	Operatori logici
Esercizio 2	Operatori di confronto
Esercizio 3	Operatori logici
Esercizio 4	Ciclo for
Esercizio 5	Ciclo for
Esercizio 6	Accumulatori
Esercizio 7	Ciclo for
Esercizio 8	Accumulatori
Esercizio 9	Accumulatori
Esercizio 10	Ciclo for
Esercizio 11	Accumulatori
Esercizio 12	Ciclo for
Esercizio 13	Ciclo while
Esercizio 14	Ciclo while
Esercizio 15	Ciclo while
Esercizio 16	Condizionali
Esercizio 17	Condizionali
Esercizio 18	Condizionali
Esercizio 19	Condizionali
Esercizio 20	Condizionali
Esercizio 21	Cicli e condizionali

Esercizio	Argomento
Esercizio 22	Cicli annidati
Esercizio 23	Ciclo for
Esercizio 24	Cicli annidati
Esercizio 25	Ciclo while

Recap 1 · `bool` e operatori di confronto

Con i tipi `number` e `string` possiamo descrivere la maggior parte degli oggetti che ci interessano.

Esempio	
Chilometri percorsi nello scorso anno	`number` - numero intero
Media dei voti	`number` - decimale
Nome e cognome	`string`
Anno di nascita	`number` - numero intero

Nel mondo reale però esiste anche un altro tipo di valore che vogliamo poter rappresentare in un linguaggio di programmazione. Ad esempio noi sappiamo valutare le seguenti espressioni:

- "il numero 3 è minore del numero 5"
- "Andrea è più alto di Marta"

Questi valori (vero o falso) sono rappresentati nei linguaggi di programmazione dal tipo `boolean`

Possiamo ottenerli effettuando operazioni **di confronto** su altri valori:

Operatore	Significato
`<` `>`	minore, maggiore
`<=` `>=`	minore o uguale, ecc.
`==` `!=`	uguale, diverso

- l'espressione `3 < 5` vale `true`

- l'espressione `"casa" == "gatto"` vale `false`

Recap 2 · Confronti e operatori logici

Operatore	Significato	Esempio (`x = 5`)
`==` `!=`	uguale, diverso	`x == 5` → `true`
`<` `<=` `>` `>=`	confronti	`x >= 6` → `false`
`&&`	**e**: entrambe vere	`x > 0 && x < 10` → `true`
`||`	**o**: almeno una vera	`x < 0 || x > 3` → `true`
`!`	**non**: nega	`!(x > 3)` → `false`

Pattern utili:

```
n % 2 == 0                  // n è pari
n % k == 0                  // n è multiplo di k
n >= 0 && n <= 10           // n compreso tra 0 e 10 (estremi inclusi)
```

Attenzione: `=` **assegna**, `==` **confronta**.

`0 <= n <= 10` non funziona: si scrive `n >= 0 && n <= 10`.

Recap 3 · Strumenti utili

Strumento	Cosa fa	Esempio
`Math.max(a, b)` / `Math.min(a, b)`	massimo / minimo	`Math.max(3, 7)` → `7`
`Math.floor(x)`	parte intera (arrotonda per difetto)	`Math.floor(2.9)` → `2`
`Math.random()`	decimale casuale in [0, 1)	`Math.floor(Math.random() * 6) + 1` → da 1 a 6
`s += x`	abbreviazione di `s = s + x`	`s -= x`, `s *= x`, ...
`x++` / `x--`	aumenta / diminuisce il valore della variabile x di 1	

Esercizio 1 · Affermazioni su *a* e *b*

Leggete due numeri `a` e `b` dall'input e stampate, **ognuna su una riga**, il valore booleano di queste affermazioni:

```
a è maggiore di b: ...
a è uguale a b: ...
a è diverso da b: ...
a è pari: ...
b è multiplo di a: ...
```

Esempio con `a = 6` e `b = 12`:

```
a è maggiore di b: false
a è uguale a b: false
a è diverso da b: true
a è pari: true
b è multiplo di a: true
```

Esercizio 2 · Confronti

Scrivere un programma che legga due numeri `N` e `M` e visualizzi, ognuno su una riga:

- il massimo tra `N` e `M`;
- il minimo tra `N` e `M`;
- se `N` è minore o uguale di `M`;
- se `N` è compreso tra 0 ed `M` (estremi inclusi);
- se `N` è divisibile per `M`.

Quali risultati sono numeri? Quali sono booleani?

Esercizio 3 · Lancio di due dadi

Simulate il lancio di **due dadi** a 6 facce: stampate i due valori, la loro somma e se è uscito un **doppio** (valore booleano).

Esempio:

```
Dado 1: 4
Dado 2: 4
Somma: 8
Doppio: true
```

Recap 4 · I cicli `for` e `while`

Entrambi **ripetono** il corpo `{ ... }` finché la condizione è vera. Questi due programmi stampano `Ciao!` **5 volte**:

```
let i;                                  // for: tutto nell'intestazione
for (i = 1; i <= 5; i = i + 1) {        // inizializzazione; condizione; aggiornamento
  console.log("Ciao!");
}
```

```
let i = 1;                              // while: i tre pezzi sono "sparsi"
while (i <= 5) {                        // condizione
  console.log("Ciao!");
  i = i + 1;                            // aggiornamento: se manca, ciclo infinito!
}
```

- `for` quando **so quante volte** ripetere; `while` quando dipende da cosa succede (es. input dell'utente);
- abbreviazione: `i++` significa `i = i + 1`, e nel `for` si può scrivere `for (let i = 1; ...)`.

Esercizio 4 · Completa la guardia (1)

Il corpo del ciclo c'è già. Completate la **condizione** (al posto di `______`) in modo che il programma stampi `Ciao!` **5 volte**.

```
let i;
for (i = 0; ______; i = i + 1) {
  console.log("Ciao!");
}
```

Esercizio 5 · Completa l'input e la guardia

Il numero di ripetizioni lo sceglie l'utente. Completate il programma (al posto di `______`) in modo che chieda `n` all'utente, lo converta in numero e stampi `Ciao!` **n volte**.

```
let n_input = ______("Quante volte?");     // chiedo il valore all'utente
let n = ______(n_input);                   // lo converto in numero
let i;
for (i = 1; ______; i = i + 1) {
  console.log("Ciao!");
}
```

Esercizio 6 · Potenze di 2

Scrivere un programma che visualizzi le potenze di 2 partendo dall'esponente 0 fino all'esponente 8, una per riga.

Esercizio 7 · Completa la guardia (2)

Il programma deve chiedere all'utente quanti numeri vuole sommare, poi chiedere i numeri uno alla volta e stampare la loro **somma**. Il corpo del ciclo c'è già: completate la **condizione** (al posto di `______`).

```
let n_input = prompt("Quanti numeri vuoi sommare?");
let n = Number(n_input);
let somma = 0;                                    // accumulatore
let i;
for (i = 1; ______; i = i + 1) {
  let x_input = prompt("Numero " + i + ":");
  let x = Number(x_input);
  somma = somma + x;
}
console.log("Somma:", somma);
```

Esercizio 8 · Somma e prodotto dei primi *n* numeri pari

Leggere un numero naturale `n` e calcolare la **somma** e il **prodotto** dei primi `n` numeri pari positivi. Stampare i due risultati alla fine.

Esempio: con `n = 4` i numeri sono `2, 4, 6, 8`, quindi la somma è `20` e il prodotto è `384`.

Esercizio 9 · Media di *n* numeri

Leggere un numero `n` e poi `n` numeri, uno alla volta. Stampare la loro **media**.

Esercizio 10 · Una riga di asterischi

Leggere un numero `n` e stampare **una sola riga** composta da `n` asterischi. Con `n = 5`:

```
*****
```

Il ciclo e la stampa ci sono già: completate il corpo (al posto di `______`) in modo da **aggiungere** un asterisco alla stringa `riga` a ogni giro.

```
let n_input = prompt("n:");
let n = Number(n_input);
let riga = "";                          // stringa vuota: l'accumulatore
let i;
for (i = 1; i <= n; i = i + 1) {
  riga = ______;
}
console.log(riga);
```

Esercizio 11 · I numeri da 1 a *n* su una riga

Leggere un numero `n` e stampare **su una sola riga** i numeri da 1 a `n`, separati da uno spazio. Con `n = 5`:

```
1 2 3 4 5
```

Suggerimento: partite dall'esercizio precedente!

Esercizio 12 · Conto alla rovescia

Completate (al posto di `_____`) in modo che il programma stampi `5`, `4`, `3`, `2`, `1` e infine `Via!`.

```
let i = 5;                          // inizializzazione
while (_____) {                     // condizione
  console.log(i);
  i = _____;                        // aggiornamento
}
console.log("Via!");
```

Esercizio 13 · Completa l'input valido

Il programma chiede un numero **positivo** e ripete la domanda finché il valore non è valido. Completate la **condizione** e la riga che **rilegge** il valore (al posto di `_____`).

```
let x_input = prompt("Inserisci un numero positivo:");
let x = Number(x_input);
while (_____) {                                  // finché NON è valido
  x_input = _____("Non valido. Riprova:");
  x = Number(x_input);
}
console.log("Hai inserito:", x);
```

Esercizio 14 · Somma fino a zero

Leggere numeri dall'input e sommarli, finché l'utente non inserisce `0`. Alla fine stampare la somma e quanti numeri sono stati inseriti (lo `0` finale escluso).

Esercizio 15 · Indovina il numero

Il programma sceglie un numero casuale tra 1 e 10. L'utente prova a indovinarlo, finché non ci riesce. Alla fine stampare in quanti tentativi ci è riuscito.

Recap 5 · `if` / `else if` / `else`

```
if (condizione1) {
  // eseguito se condizione1 è true
} else if (condizione2) {
  // eseguito se condizione1 è false e condizione2 è true
} else {
  // eseguito se nessuna delle precedenti è true
}
```

- la condizione è un'espressione che vale `true` o `false` (ad esempio `eta >= 18`);
- viene eseguito **un solo** blocco: il primo la cui condizione è vera;
- `else if` e `else` sono **facoltativi**;
- l'ordine conta: le condizioni vengono controllate dall'alto verso il basso.

```
let voto = 24;
if (voto >= 18) {
  console.log("Esame superato");
} else {
  console.log("Esame non superato");
}
```

Esercizio 16 · Pari o dispari

Leggere un numero intero e stampare `"pari"` oppure `"dispari"`.

Esercizio 17 · Radice quadrata

Dato un numero reale in input, mostrare il valore della sua radice quadrata.

Fare inizialmente il controllo che il numero sia **non negativo**, dato che la radice quadrata è definita solo per argomenti reali maggiori o uguali a zero. Altrimenti stampare un messaggio di errore.

Esercizio 18 · Massimo di tre numeri

Leggere tre numeri e stampare il più grande, **senza** usare `Math.max`.

Esercizio 19 · Voto e giudizio

Leggere un voto da 0 a 30 e stampare un giudizio:

Voto	Giudizio
minore di 18	insufficiente
da 18 a 23	sufficiente
da 24 a 27	buono
da 28 a 30	ottimo
fuori da 0–30	voto non valido

Esercizio 20 · Costo di una chiamata

Leggere la durata in minuti (numero reale) di una chiamata telefonica e visualizzare il costo, sapendo che:

- lo scatto alla risposta costa 0.18 €;
- il primo minuto è gratuito;
- ogni minuto successivo al primo costa 0.16 €.

Se la durata è negativa, stampare un messaggio di errore.

Esercizio 21 · Somma di multipli

Calcolare la somma di tutti i numeri da 1 a `n` che sono multipli di 3 **oppure** di 5.

Esempio: con `n = 10` si sommano `3, 5, 6, 9, 10` → `33`.

Esercizio 22 · Tabellina

Leggere un numero `n` e stampare la sua tabellina (da `n · 1` a `n · 10`) **in un'unica stringa**.

Se `n` non è un numero intero compreso tra 1 e 10, stampare un messaggio di errore.

Esempio con `n = 5`: `5 10 15 20 25 30 35 40 45 50`

Scrivere due versioni: una con `for`, una con `while`.

Esercizio 23 · Polinomio

Leggere un numero reale `x` e un numero naturale `n` (con `n ≥ 1`) e calcolare

`x + x² + x³ + ... + xⁿ`

Esercizio 24 · Coppie con somma *n*

Realizzare un programma che stampi tutte le coppie di numeri interi positivi (diversi tra loro) la cui somma è un numero naturale `n` a scelta, maggiore di 2.

Ad esempio, con `n = 3` le coppie da visualizzare sono:

```
1, 2
2, 1
```

Accertarsi inizialmente che `n` sia effettivamente un numero naturale maggiore di 2. Fare uso del ciclo `for`.

Esercizio 25 · Primi *N* pari dopo *A*

Calcolare la somma e la media dei primi `N` numeri interi positivi **pari** successivi a un numero intero positivo `A`. Sia `N` che `A` devono essere letti da tastiera.

Accertarsi inizialmente che `N` ed `A` siano numeri interi positivi.

Esempio: con `A = 5` e `N = 3` i numeri sono `6, 8, 10`: somma `24`, media `8`.

Recap 6 · && e || con valori non booleani

In JavaScript `&&` e `||` non restituiscono sempre `true` o `false`: restituiscono **uno dei due operandi**. Per decidere, ogni valore viene letto come vero o falso:

- **falsi** (*falsy*): `false`, `0`, `""` (stringa vuota), `NaN`, `null`, `undefined`

- **veri** (*truthy*): tutto il resto, compresi `5`, `-1`, `"0"` e `"false"` (stringhe non vuote)

Le regole, con valutazione da sinistra a destra:

Operatore	Se il primo operando è...	Risultato
`a || b`	vero	`a` (b non viene nemmeno valutato)
`a || b`	falso	`b`
`a && b`	falso	`a` (b non viene nemmeno valutato)
`a && b`	vero	`b`

In pratica, `||` restituisce il **primo operando vero** (o l'ultimo, se sono tutti falsi), mentre `&&` restituisce il **primo operando falso** (o l'ultimo, se sono tutti veri).

Cosa stampano queste istruzioni?

```
console.log(5 || true);
console.log(5 || false);
console.log(0 || false);
console.log(0 || true);
console.log("0" || true);
console.log("0" || false);
console.log(5 && true);
console.log(5 && false);
console.log(0 && true);
console.log(0 && false);
console.log("0" && true);
```


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