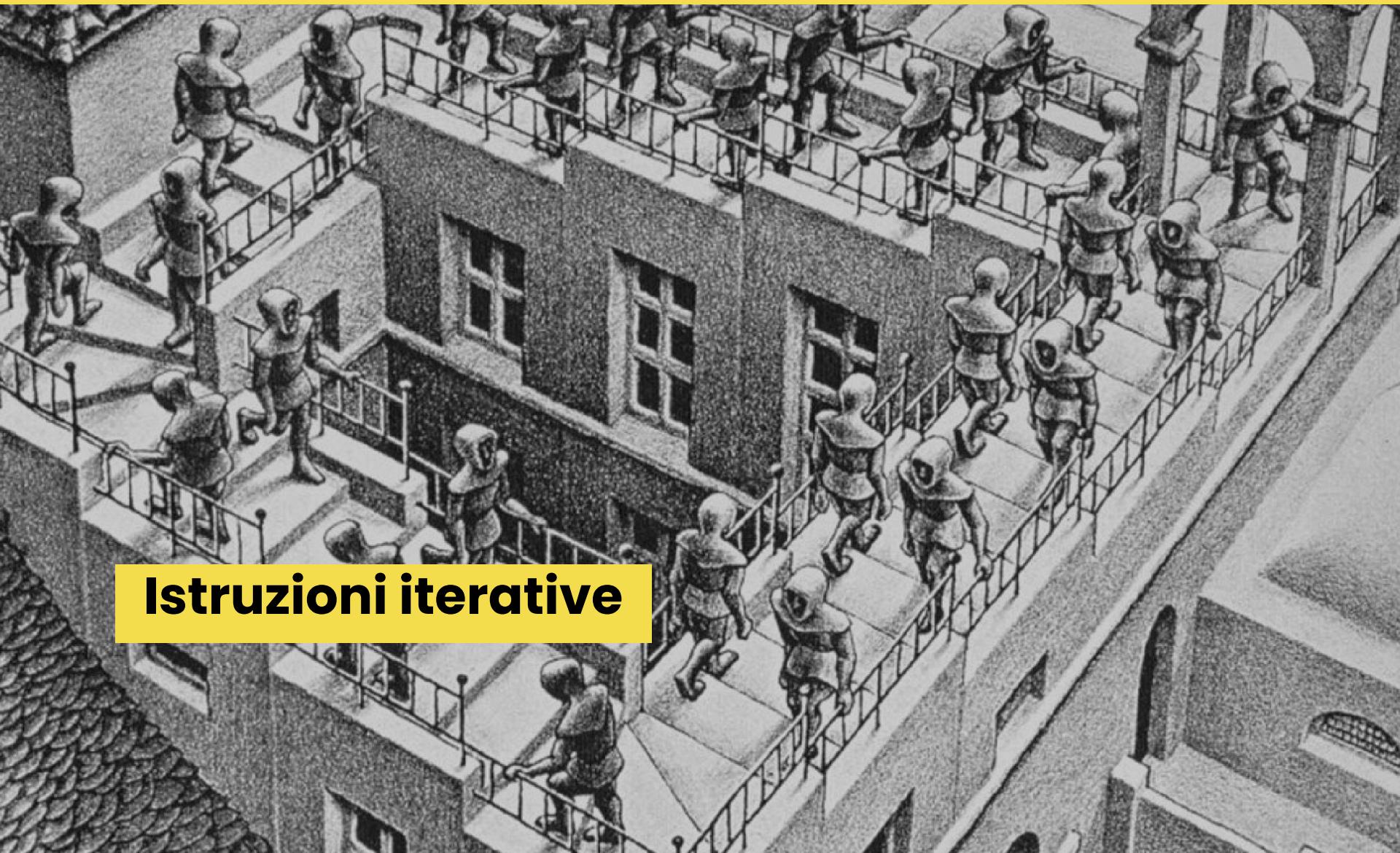


# Istruzioni iterative



# Esempio

Scrivere un programma che visualizzi tutti i numeri interi da uno a 10.

```
console.log(1);
console.log(2);
console.log(3);
console.log(4);
console.log(5);
console.log(6);
console.log(7);
console.log(8);
console.log(9);
console.log(10);
```

# Esempio

Scrivere un programma che visualizzi tutti i numeri interi da uno a 20.

```
console.log(1);
console.log(2);
console.log(3);
console.log(4);
console.log(5);
console.log(6);
console.log(7);
console.log(8);
console.log(9);
console.log(10);
```

```
console.log(11);
console.log(12);
console.log(13);
console.log(14);
console.log(15);
console.log(16);
console.log(17);
console.log(18);
console.log(19);
console.log(20);
```

# Esempio

Scrivere un programma che visualizzi tutti i numeri interi da uno a 1000.



L'idea di usare un programma per risolvere un problema consiste nel lavorare di meno, non di più.

# Istruzioni iterative (cicli)

- Molti problemi richiedono un calcolo che deve essere ripetuto più volte per ottenere il risultato finale.
- Le istruzioni iterative consentono di indicare la necessità di ripetere un calcolo più volte.

# Tipi di istruzioni iterative

## Determinate

il numero di ripetizioni è **noto a priori**.

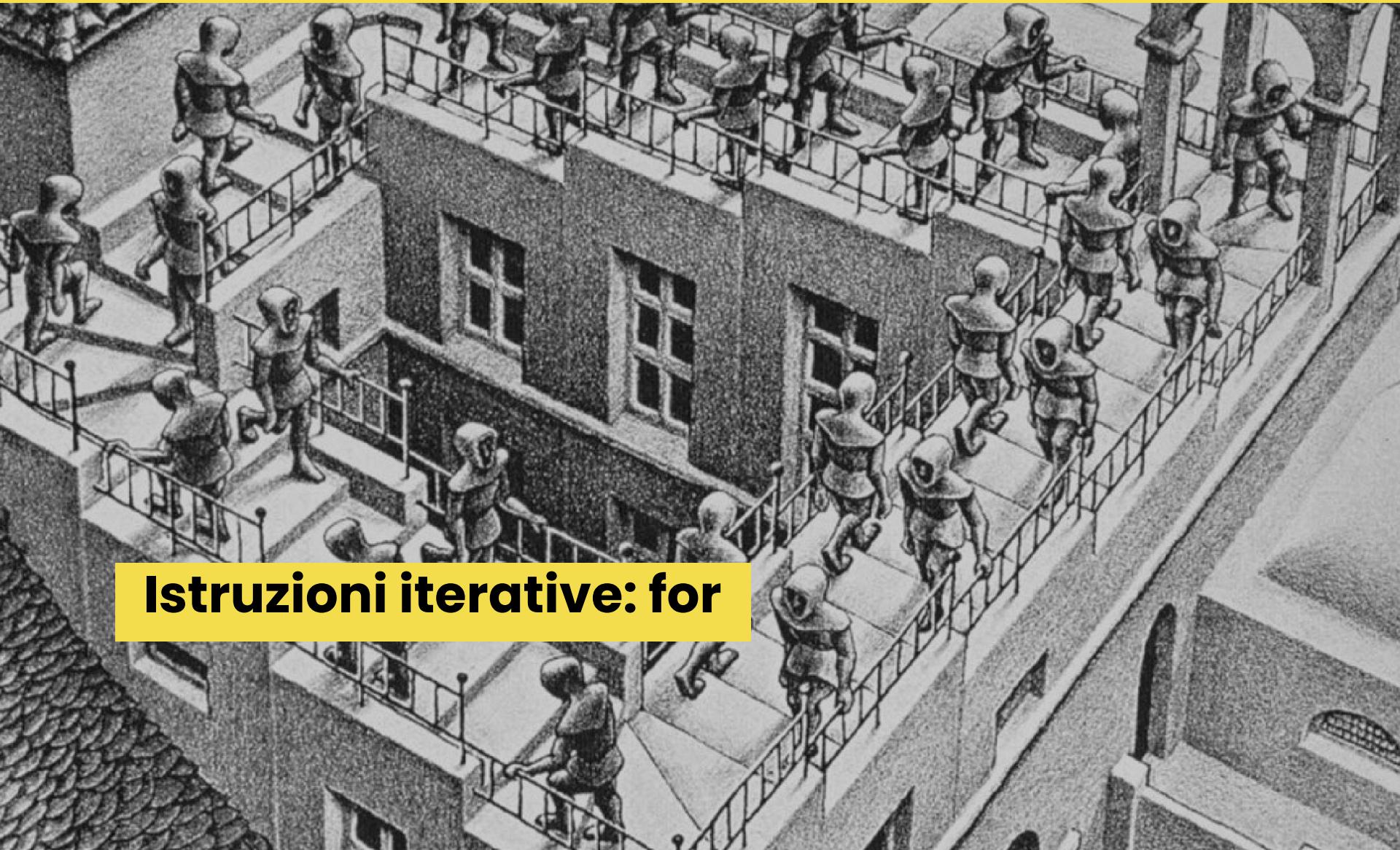
- fai **10 giri** del parco di corsa
- leggi dall'input **k** numeri
- genera **5 numeri** casuali

## Indeterminate

il numero di ripetizioni **non è noto a priori**.

- **finchè non** sei sazio mangia
- ripeti l'esame di Fondamenti di Informatica **fino a che** il voto  $\geq 18$
- continua a leggere dall'input **finchè non** leggi un numero negativo

### Istruzioni iterative: for



## Istruzione for

**Obiettivo:** gestire le iterazioni **determinate** in modo diretto.

Utilizziamo una variabile contatore per contare quante volte viene eseguito il ciclo

# Sintassi dell'istruzione for

```
for (inizializzazione; condizione; incremento) {  
    // istruzioni da ripetere  
}
```

- **inizializzazione**: assegna un valore iniziale alla variabile contatore
- **condizione**: esprime la condizione di uscita dal ciclo
- **incremento**: modifica il valore della variabile contatore

Esempio: *risolvere il problema di visualizzare i numeri da 1 a 1000*

```
for (let i = 0; i < 1000; i++) {  
    console.log(i);  
}
```

# Semantica dell'istruzione for

```
for (inizializzazione; condizione; incremento) {  
    // istruzioni da ripetere  
}
```

1. viene eseguita l'**inizializzazione** (una sola volta)
2. viene valutata la **condizione**:
  - se è **VERA**:
    - a. vengono **eseguite le istruzioni** nel corpo del ciclo
    - b. viene **eseguito l'incremento**
    - c. si **ritorna al punto 2**
  - se è **FALSA**:  
**si esce dal ciclo**

## Ruolo del contatore

- Il contatore `i` è una variabile che **conta** le iterazioni.
- Il contatore `i` è una variabile che **controlla** la condizione di **uscita** dal ciclo.
- Usando il `for` variamo in modo automatico il valore del contatore

```
for (i = 10; i >= 1; i=i-1) {  
    console.log(i);  
}
```

### output

`i = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1`

```
for (i = -4; i <= 4; i=i+2) {  
    console.log(i);  
}
```

### output

`i = -4, -2, 0, 2, 4`

# Abbreviazioni

Alcune utili funzioni per abbreviare istruzioni di assegnamento con variabili numeriche

Data una variabile numerica x:

`x = x+1;` si può abbreviare con  
`x++ ;`

`x = x-1;` si può abbreviare con `x-`  
`- ;`

Date due variabili numeriche x e y:

`x = x + y;` si può abbreviare con  
`x+=y;`

`x = x - y;` si può abbreviare con  
`x-=y;`

`x = x * y;` si può abbreviare con  
`x*=y;`

`x = x / y;` si può abbreviare con  
`x/=y;`

`x = x % y;` si può abbreviare con  
`x%=y;`

# Esempio

Scrivere una funzione che calcoli e restituisca la somma di tutti i numeri interi dispari compresi tra 1 ed n (dove n e' un parametro della funzione)

## Esempio – Soluzione

Scrivere una funzione che calcoli e restituisca la somma di tutti i numeri interi dispari compresi tra 1 ed n (dove n e' un parametro della funzione)

```
let somma = 0;  
const n = 10;  
for (let i = 1; i <= n; i+=2) {  
    somma += i;  
}  
  
console.log(somma);
```

## Esempio - Soluzione 2

Scrivere una funzione che calcoli e restituisca la somma di tutti i numeri interi dispari compresi tra 1 ed n (dove n e' un parametro della funzione)

```
let somma = 0;  
const n = 10;  
for (let i = 1; i <= n; i++) {  
    if (i % 2 !== 0) {  
        somma += i;  
    }  
}  
  
console.log(somma);
```

# Esercizio

Scrivere un codice che generi  $n$  numeri casuali e restituisca il loro *massimo*.

## Esercizio – Soluzione

Scrivere un codice che generi  $n$  numeri casuali e restituisca il loro *massimo*.

```
let max = 0;
const n = 10;
for (let i = 0; i < n; i++) {
    let numero = Math.floor(Math.random() * 100);
    if (numero > max) {
        max = numero;
    }
}

console.log(max);
```

## Esercizio

Scrivere un codice che generi  $n$  numeri casuali e restituisca la loro *media*.

## Esercizio – Soluzione

Scrivere un codice che generi  $n$  numeri casuali e restituisca la loro *media*.

```
let somma = 0;
let n = 10;
for (let i = 0; i < n; i++) {
    let numero = Math.floor(Math.random() * 100);
    somma += numero;
}

let media = somma / n;
console.log('La media è: ' + media);
```

## Esercizio

Scrivere un codice che generi  $n$  numeri casuali e restituisca il *minimo* e il *massimo*.

```
let min = 100;
let max = 0;
let n = 10;
for (let i = 0; i < n; i++) {
    let numero = Math.floor(Math.random() * 100);
    if (numero > max) {
        max = numero;
    }
    if (numero < min) {
        min = numero;
    }
}

console.log('Il minimo è: ' + min);
console.log('Il massimo è: ' + max);
```

### Istruzioni iterative: WHILE

## Istruzione indeterminata

Il **calcolo** viene ripetuto **finché una condizione è vera**.

Esempio:

- finché non sei sazio mangia
- ripeti l'esame di Fondamenti di Informatica fino a che il voto  $\geq 18$
- continua a leggere dall'input finché non leggi un numero negativo

# Istruzione while

Sintassi dell'istruzione while

```
while (condizione) {  
    // istruzioni da ripetere  
}
```

ripete l'esecuzione delle istruzioni finché la condizione è vera.

Il corpo del ciclo può essere un blocco di istruzioni

```
while (condizione) {  
    istruzione_1  
    istruzione_2  
    ...  
    istruzione_n  
}
```

## Istruzioni FOR vs WHILE

- **for**: utilizzato per iterazioni **determinate**
- **while**: utilizzato per iterazioni **indeterminate**

Con il while è ancora possibile testare quante iterazioni abbiamo fatto, quindi di fatto è possibile usare un while come un for.

Le differenze sono:

- nella sintassi dobbiamo ricordarci di inizializzare il contatore e di incrementarlo, altrimenti generiamo un loop infinito!
- nell'uso con cui sono stati pensati

## Esempio

```
let i = 0;  
while (i < 10) {  
    console.log(i);  
    i++;  
}
```

```
let i = 0;  
for (i = 0; i < 10; i++) {  
    console.log(i);  
}
```

I due esempi precedenti sono equivalenti, ma **il while è più flessibile e può essere utilizzato per iterazioni indeterminate.**

## Iterazioni indeterminate - Espressione

Nota bene: l'espressione viene valutata prima di ogni iterazione.

Nel caso in cui si voglia valutare l'espressione alla fine dell'iterazione, è possibile utilizzare l'istruzione `do...while`.

Esempio:

```
do {  
    // istruzioni  
} while (condizione);
```

Nota bene: le istruzioni vengono eseguite almeno una volta.

## Iterazione indeterminata - Do...while

```
do istruzione;  
while (condizione);
```

Semantica:

1. Viene eseguito il corpo dell'istruzione
2. Viene valutata l'espressione
3. Se è vera si ritorna al do e si ripete il ciclo
4. Se è falsa si passa alla prima istruzione dopo il do  
while

# Esempio

Scrivere un programma che generi un numero casuale tra -1 e 1 finché non viene generato un numero positivo, restituendo quindi il numero positivo ottenuto e il numero di tentativi effettuati.

```
let tentativi = 0;
let numero = 0;
do {
    numero = Math.random() * 2 - 1;
    tentativi++;
} while (numero <= 0);

console.log('Numero positivo: ' + numero);
console.log('Tentativi: ' + tentativi);
```

**Riscrivi l'esercizio precedente senza il costrutto do...while**

```
let tentativi = 0;
let numero = 0;
while (numero <= 0) {
    numero = Math.random() * 2 - 1;
    tentativi++;
}

console.log('Numero positivo: ' + numero);
console.log('Tentativi: ' + tentativi);
```

La differenza è che con il while il corpo del ciclo viene eseguito solo dopo aver valutato la condizione.

## Come si esce da un ciclo?

Normalmente:

- Un ciclo viene ripetuto finché il controllo non diventa falsa

Ma c'è un altro modo:

- L'istruzione `break` consente di uscire da un ciclo immediatamente

## Esempio

Scrivere una funzione che generi al massimo `n` numeri casuali fino a che non ne

trova uno maggiore di una `soglia`

La funzione deve restituire quanti numeri casuali sono stati generati.

## Esempio – Soluzione

```
let soglia = 50;
let n = 100;
let tentativi = 0;
for (let i = 0; i < n; i++) {
    let numero = Math.floor(Math.random() * 100);
    tentativi++;
    if (numero > soglia) {
        break;
    }
}

console.log('Tentativi: ' + tentativi);
```