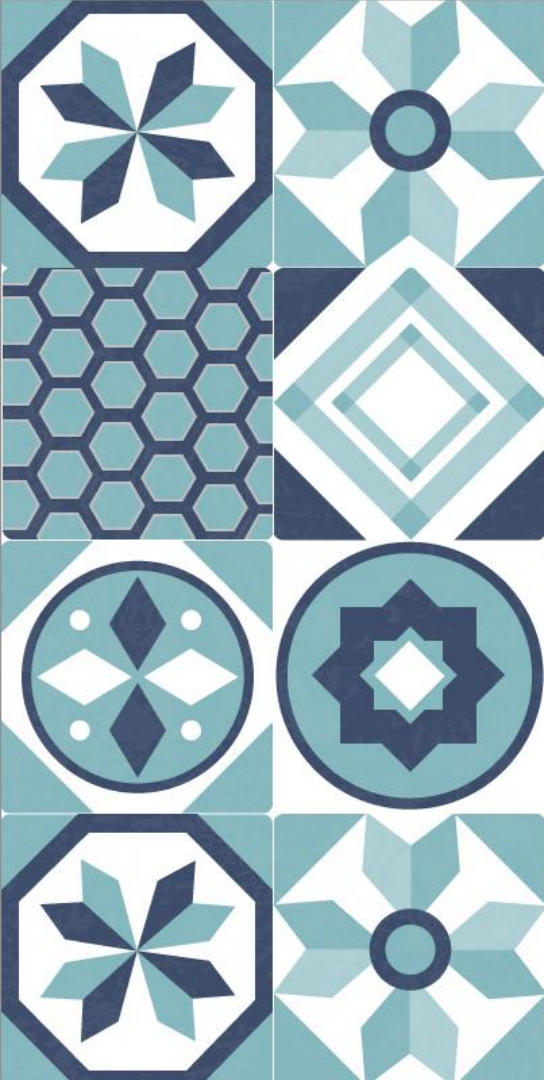


FONDAMENTI DI INFORMATICA

Alma Artis
Francesca Pratesi (ISTI, CNR)

Iterazioni indeterminate

ESERCIZI

- Scrivere una funzione con due parametri numerici interi n ed m. La funzione deve verificare se n è divisibile per almeno un intero tra 2 ed m. In caso positivo la funzione deve restituire true, altrimenti false. Si invochi la funzione precedentemente definita, passando come parametri tre coppie di valori scelti opportunamente. Si stampi quindi il risultato ottenuto nei tre casi.

- Scrivere una funzione che prende un parametro numerico intero n. La funzione deve generare n numeri casuali e restituire true se almeno un numero è maggiore di 0,7, false altrimenti. Si invochi la funzione precedentemente definita.

- Scrivere una funzione che prende due parametri numerici interi n ed m. La funzione deve chiedere n numeri all'utente e calcolare quanti dei numeri letti sono minori di m. Si invochi la funzione precedentemente definita.

ESERCIZI (2)

- Il fattoriale di un numero intero n > 1, indicato con n!, è uguale al prodotto dei numeri interi da 1 ad n.

  Cioè: $n!=1*2*3*...*n$

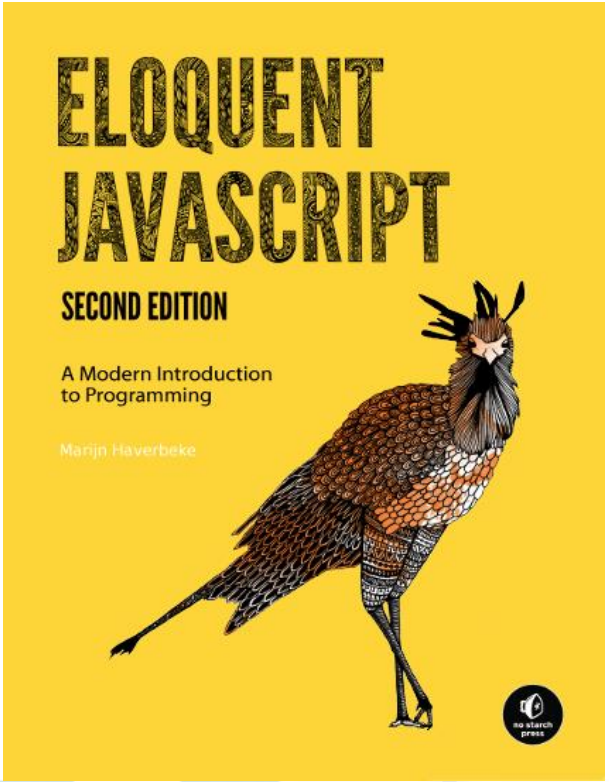
  Ad esempio, se n = 4, n! = 1 * 2 * 3 * 4 = 24.

  Si scriva una funzione con un parametro n che calcoli e restituisca in output il valore del fattoriale di n.

  Invocare la funzione definita al punto precedente usando come parametro un numero intero arbitrario. Si stampi quindi il risultato ottenuto.

ISTRUZIONI ITERATIVE

LIBRI E RIFERIMENTI



- Capitolo 2

Eloquent Javascript – Second Edition
Marijn Haverbeke

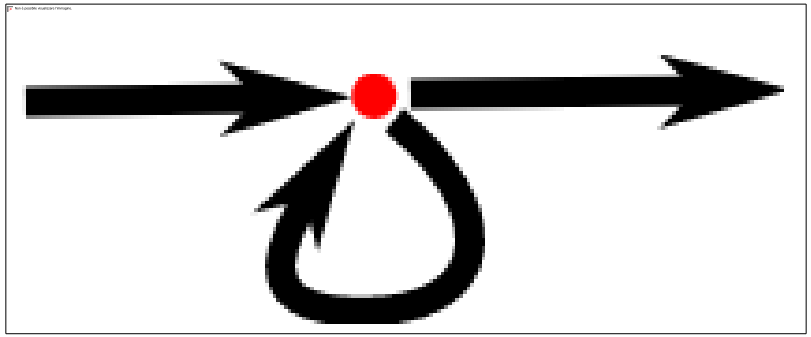
Available here: http://eloquentjavascript.net/

ISTRUZIONI ITERATIVE

- 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.

ISTRUZIONI ITERATIVE / CICLI

Un modo per eseguire un pezzo di codice (un insieme di istruzioni) ripetutamente



ciclo
(o loop)

TIPI DI ISTRUZIONI ITERATIVE

DETERMINATE

INDETERMINATE

ITERAZIONE DETERMINATA

Il calcolo viene ripetuto un numero fissato di volte.
Es.:

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

ITERAZIONE INDETERMINATA

Il calcolo viene ripetuto finchè una condizione è vera.
Es.:

- finchè non sei sazio mangia
- leggi un numero dall'input finchè non trovi un numero negativo
- ripeti l'esame di Fondamenti di Informatica fino a che il voto >= 18

ISTRUZIONI ITERATIVE: WHILE

ISTRUZIONE WHILE

Sintassi

```
while (espressione)
    istruzione_ciclo;
```

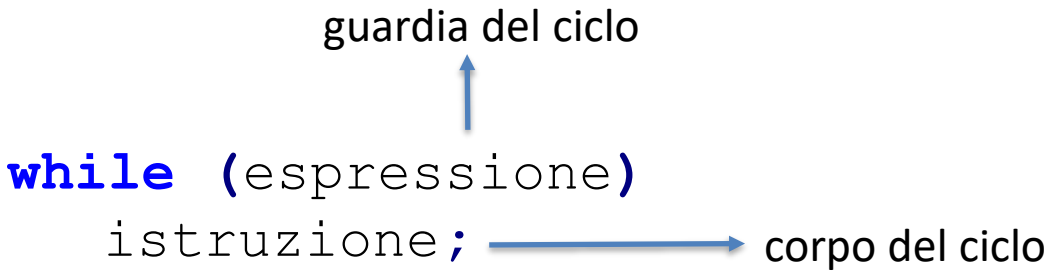
ripete istruzione_ciclo finché l'espressione è vera

Il corpo del ciclo può essere un blocco (in questo caso ci sono più istruzioni che vengono ripetute)

```
while (espressione){
    istruzione_blocco1;
    istruzione_blocco2;
    // etc…
    istruzione_bloccoN;
}
```

ISTRUZIONE WHILE

Sintassi



Semantica:

1) Viene valutata l'espressione
2) Se è vera esegue il corpo del ciclo e ritorna al punto 1
3) Se è falsa si passa alla prima istruzione dopo il while

ISTRUZIONE FOR: ESEMPIO

```
var condizione = true, numero;
while (condizione){
  numero = Number(prompt('inserisci un numero'));
  if (numero<0) condizione=false;
}
```

ISTRUZIONE FOR VS WHILE

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 i contatore e di incrementarlo, altrimenti generiamo un loop infinito!
- nell'uso con cui sono stati pensati

ISTRUZIONE FOR VS WHILE

Esempio

```
var i, k = 10;
for (i=1; i<=k; i++){
  //istruzioni del ciclo
}
```

```
var i=1, k = 10;
while (i<=k){
  //istruzioni del ciclo
  i++;
}
```

ITERAZIONE INDETERMINATA - ESPRESSIONE

Nota bene: l'espressione viene valutata all'**inizio** di ogni iterazione

ITERAZIONE INDETERMINATA – DO WHILE

```
do
    istruzione;
while (espressione);
```

Con il do-while l'espressione viene valutata alla **fine** di ogni iterazione

Nota bene: il corpo del while viene eseguito almeno una volta

ITERAZIONE INDETERMINATA – DO WHILE

```
do istruzione;
while (espressione);
```

Semantica:

1) Viene eseguito il corpo del ciclo
2) Viene valutata l'espressione
3) Se è vera si ritorna al punto 1
4) Se è falsa si passa alla prima istruzione dopo il do-while

ESEMPIO

Scrivere una funzione che generi un numero casuale tra -1 e 1 finché non viene generato un numero positivo, restituendo quindi il numero positivo ottenuto.

ESEMPIO

```
function random_positivo(){
var numero;
do
numero =2*Math.random()-1;
while(numero<=0);

return numero;
}
```

ESEMPIO 2 (CONTROLLA LE DIFFERENZE)

```
function random_positivo_while(){
var numero;
numero =2*Math.random()-1;
while(numero <=0)
numero =2*Math.random()-1;

return numero;
}
```

ESEMPIO

Scrivere una funzione che legga in input dei valori numerici, arrestandosi quando incontra il valore di un terminatore (parametro della funzione). La funzione deve restituire quanti numeri sono stati letti prima del terminatore.

ESEMPIO

```
function lunghezza_sequenza(t){
var numero;
var l =0;//quanti numeri ho letto diversi da t
do{
numero = Number(prompt());//leggo un numero
l++;//aggiorno il contatore
}while(numero!=t);//itero finche' non trovo il terminatore

/*quando esco dal ciclo do-while so che l'ultimo numero letto
e' il terminatore, quindi la lunghezza della sequenza va
decrementata di 1*/
return(l-1);
}
```

ESEMPIO

```
function loop_terminatore_do_while(terminatore){
    var quanti = -1;
    var numero;
    do{
        numero = prompt("Inserisci un numero: ");
        quanti++;
    } while (numero!=terminatore)
    return quanti;
}

var term = prompt("Inserisci il terminatore: ");
console.log(loop_terminatore_do_while(term));
```

```
function loop_terminatore_while(terminatore){
    var numero;
    numero = prompt("Inserisci un numero: ");
    var quanti = 0;
    while (numero!=terminatore){
        numero = prompt("Inserisci un numero: ");
        quanti++;
    }
    return quanti;
}

var term = prompt("Inserisci il terminatore: ");
console.log(loop_terminatore_while(term));
```

COME SI ESCE DA UN CICLO

- Normalmente:
  - Un ciclo viene ripetuto finché la guardia 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

```
function maggiore_soglia_quanti(n,soglia){
    var i;
    var numero;
    for(i =1;i<=n;i++){
    numero=Math.random();
    if(numero>soglia)
        break;
    }

    return i;
}
```