

# Le strutture iterative

*do-while; while; for;*

# ITERAZIONE (1/3)

Supponiamo di dovere prelevare un libro da uno scaffale di una biblioteca. Se non conosciamo in quale scaffale è contenuto il libro, possiamo seguire questo algoritmo di ricerca:

```
considera il primo scaffale  
se in esso è contenuto il libro  
  allora estrailo  
  altrimenti considera lo scaffale successivo
```

Sorge spontaneamente il problema di sapere quante volte è necessario considerare scaffali, cioè quante volte scrivere «se... allora... altrimenti». Le strutture viste finora non sono sufficienti ed è necessario aggiungere un nuovo costrutto, la **struttura iterativa**, con la quale è possibile stabilire un ciclo per eseguire tante volte un insieme di istruzioni.

## ITERAZIONE (2/3)

L'algoritmo precedente diventa:

considera il primo scaffale

ripeti

se in esso è contenuto il libro

allora estrailo

altrimenti considera lo scaffale successivo

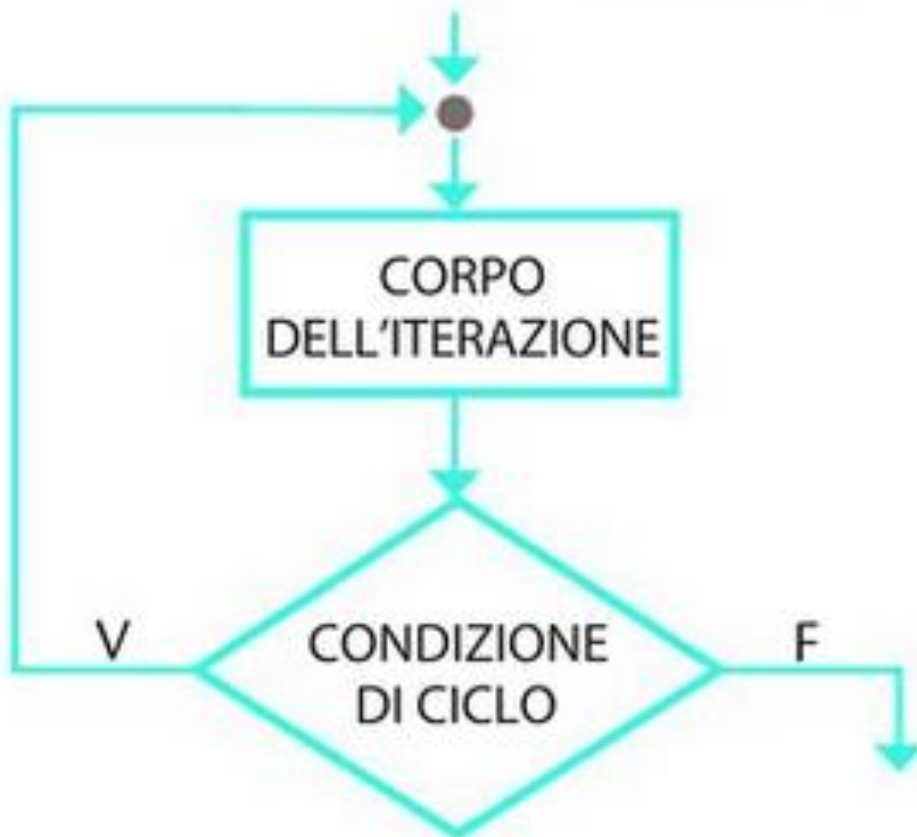
finché ci sono scaffali e non hai trovato il libro

*CORPO DEL CICLO*

Le istruzioni che sono contenute tra le parole chiave «ripeti» e «finché» formano il **corpo dell'iterazione** e saranno ripetute tante volte in fase di esecuzione dell'algoritmo.

# ITERAZIONE 3/3

A livello di algoritmo possiamo esprimere la struttura iterativa con due modalità: con il controllo in coda (**postcondizionale**) e con il controllo in testa (**precondizionale**).



Il diagramma a blocchi per la **struttura iterativa postcondizionale** (cioè con controllo in coda)



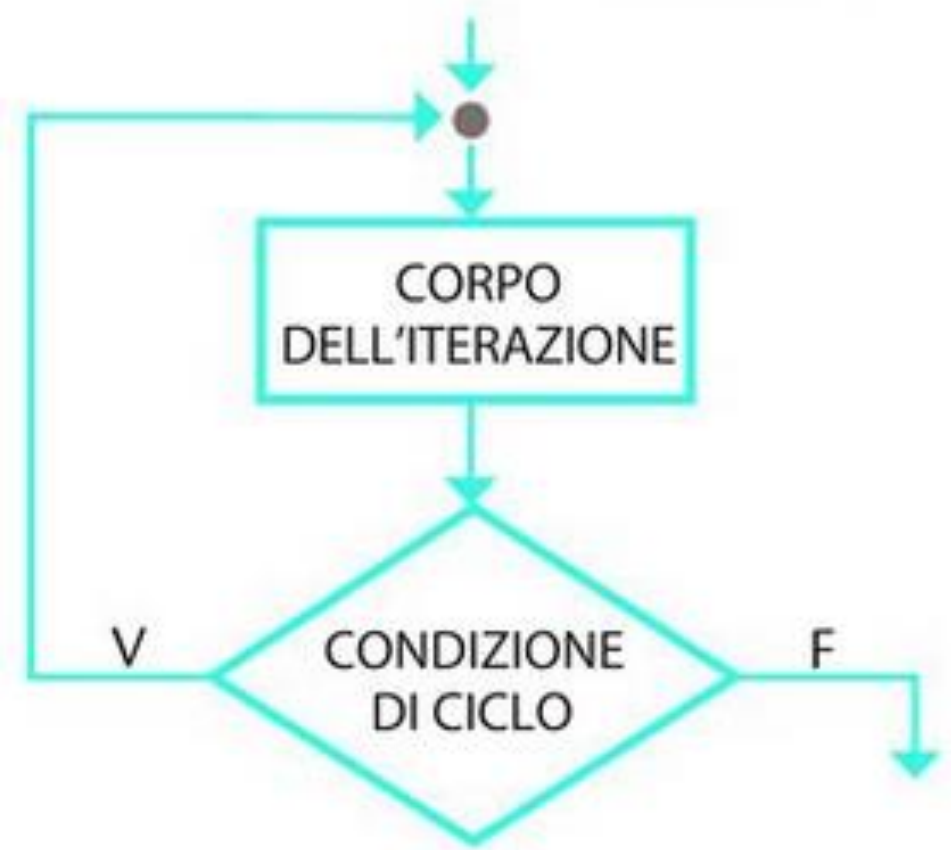
Il diagramma a blocchi per la **struttura iterativa precondizionale** (cioè con controllo in testa)

# Ciclo post-condizione

RIPETI

<corpo dell'iterazione>

FINCHÉ <condizione>



*Prima* sono eseguite le istruzioni che formano il corpo dell'iterazione e *dopo* è eseguito il controllo per stabilire se ripetere il corpo dell'iterazione (ciclo).

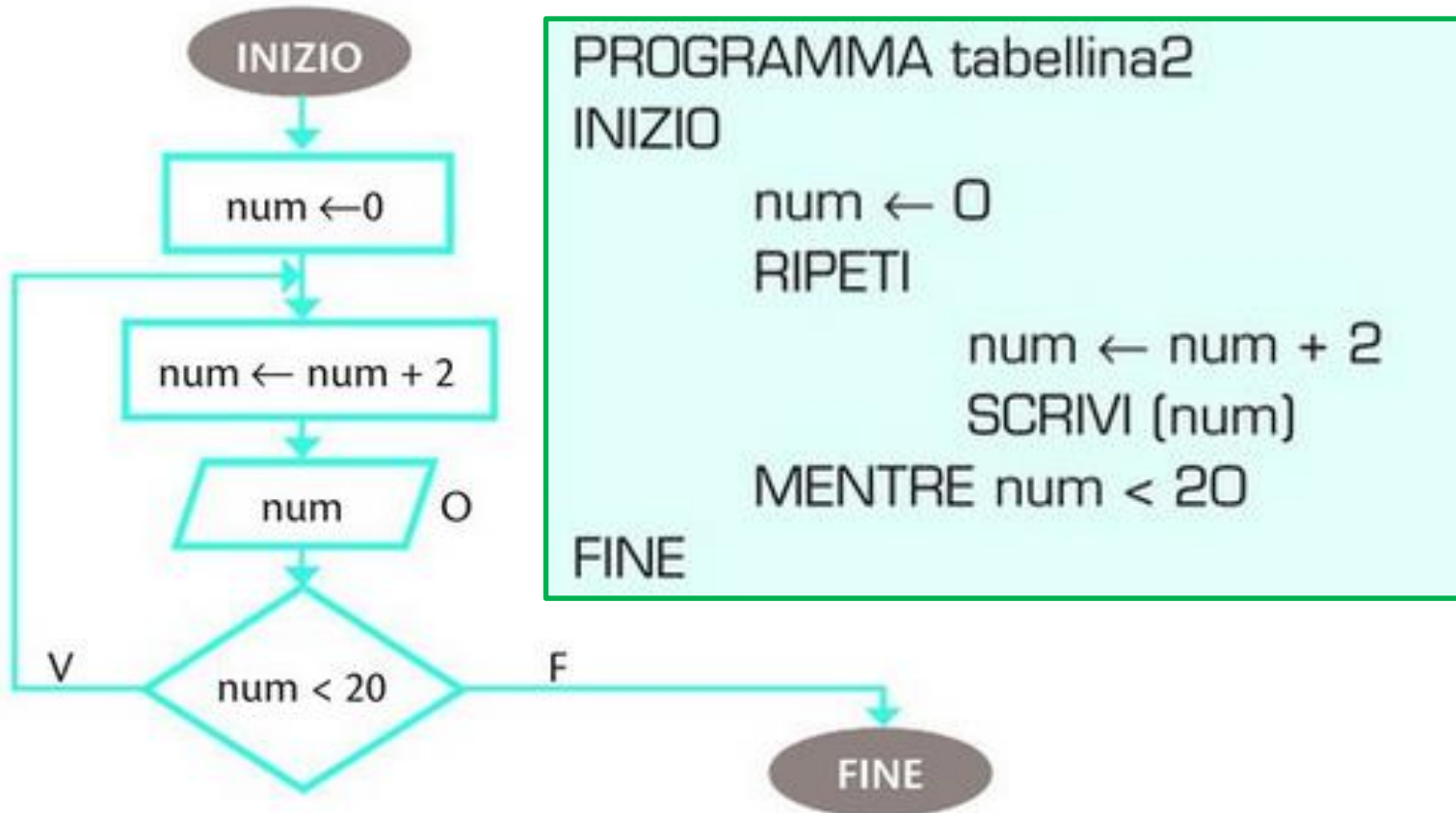
Se la condizione di uscita del ciclo è verificata (cioè è vera), allora il ciclo viene ripetuto, altrimenti l'esecuzione prosegue con la prima istruzione che si trova sul ramo del falso.

Il corpo dell'iterazione è eseguito *almeno una volta*.



# Esercizio: *iterazione post-condizione*

Scrivere una programma che stampi in uscita una sequenza di numeri che rappresentano la tabellina del 2, a partire dal numero 2 sino al 20



```
int num;  
num = 0;  
  
//ELABORAZIONE  
do  
{  
    num = num + 2;  
    System.out.print("Ecco il valore  
di num:"+num);  
}while (num<20);
```

# Esercizio: Ciclo post-condizione

**TESTO:** Scrivere un programma che controlla i dati messi in input e valida questi solo se maggiori di 0 e minori di 11

```
import java.util.*;

public class CicloPrecondizionale {
    public static void main(String[] args) {
        int num;
        //istanzia un oggetto lettore di tipo Scanner
        Scanner in = new Scanner(System.in);

        num = 0;

        //ELABORAZIONE
        do{
            //Chiede di inserire un numero compreso tra 0 e 11
            System.out.print("Inserisci un numero intero (n>0 e <11): ");
            num = in.nextInt();
        } while( !((num>0) && (num<11)) );

        System.out.println("Ecco il numero inserito: "+num);
    } //main
} //class
```

# Esercizio: Ciclo post-condizione

**TESTO:** Scrivere un programma che, utilizzando il ciclo do-while, calcola e stampi la media di n numeri interi in input

```
public static void main(String[] args) {
    int num, conta, n, somma;
    float media;
    //istanzia un oggetto lettore di tipo Scanner
    Scanner in = new Scanner(System.in);

    System.out.print("Quanti numeri vuoi elaborare(Inserisc N): ");
    n = in.nextInt();

    somma = 0;
    conta = 1;
    do{
        System.out.print("Inserisci un numero intero: ");
        num = in.nextInt();
        somma = somma + num;
        conta++;
    }while(conta<=n);

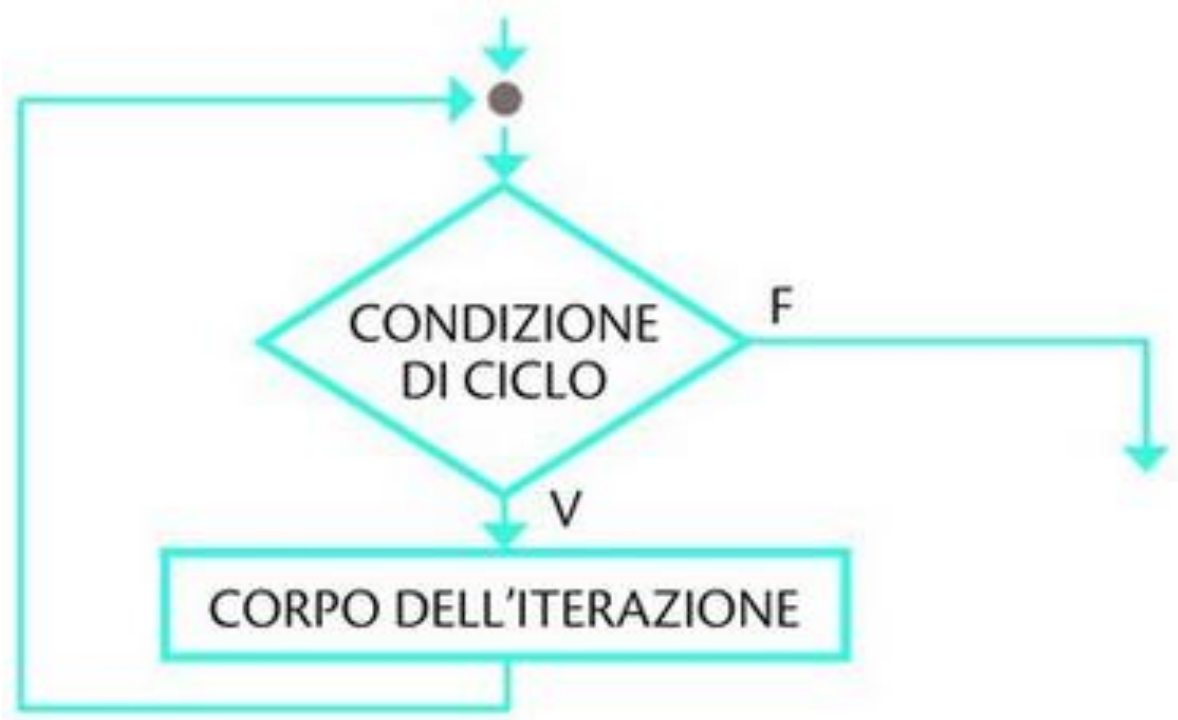
    System.out.println("La SOMMA dei numeri inseriti è "+somma);
    media = (float) somma/n;
    System.out.println("La MEDIA dei numeri inseriti è "+media);
} //main
```

```
Quanti numeri vuoi elaborare(Inserisc N): 4
Inserisci un numero intero: 1
Inserisci un numero intero: 2
Inserisci un numero intero: 3
Inserisci un numero intero: 4
La SOMMA dei numeri inseriti è 10
La MEDIA dei numeri inseriti è 2.5
```



# Ciclo pre-condizione

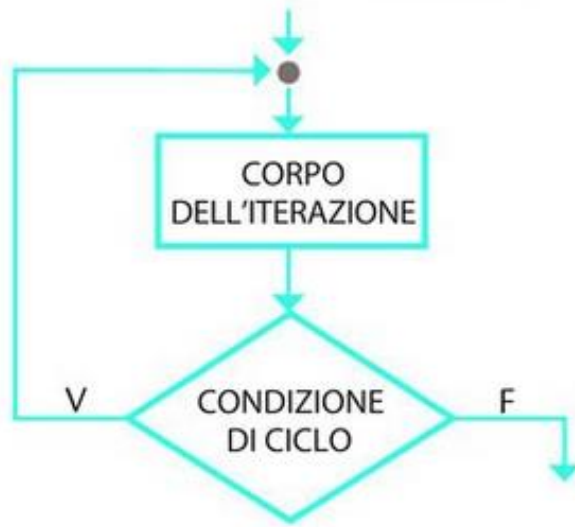
MENTRE <condizione> ESEGUI  
<corpo dell'iterazione>



In questo caso prima viene eseguito il controllo per ripetere il ciclo e, se il controllo risulta *vero*, è eseguito il corpo dell'iterazione. Quando la condizione di ciclo diventa falsa, allora il ciclo termina e l'esecuzione prosegue con la prima istruzione che si trova sul ramo del *falso*.

A differenza del ciclo precedente, il corpo dell'iterazione potrebbe non essere mai eseguito: questo accade quando la condizione di ciclo è falsa in partenza.

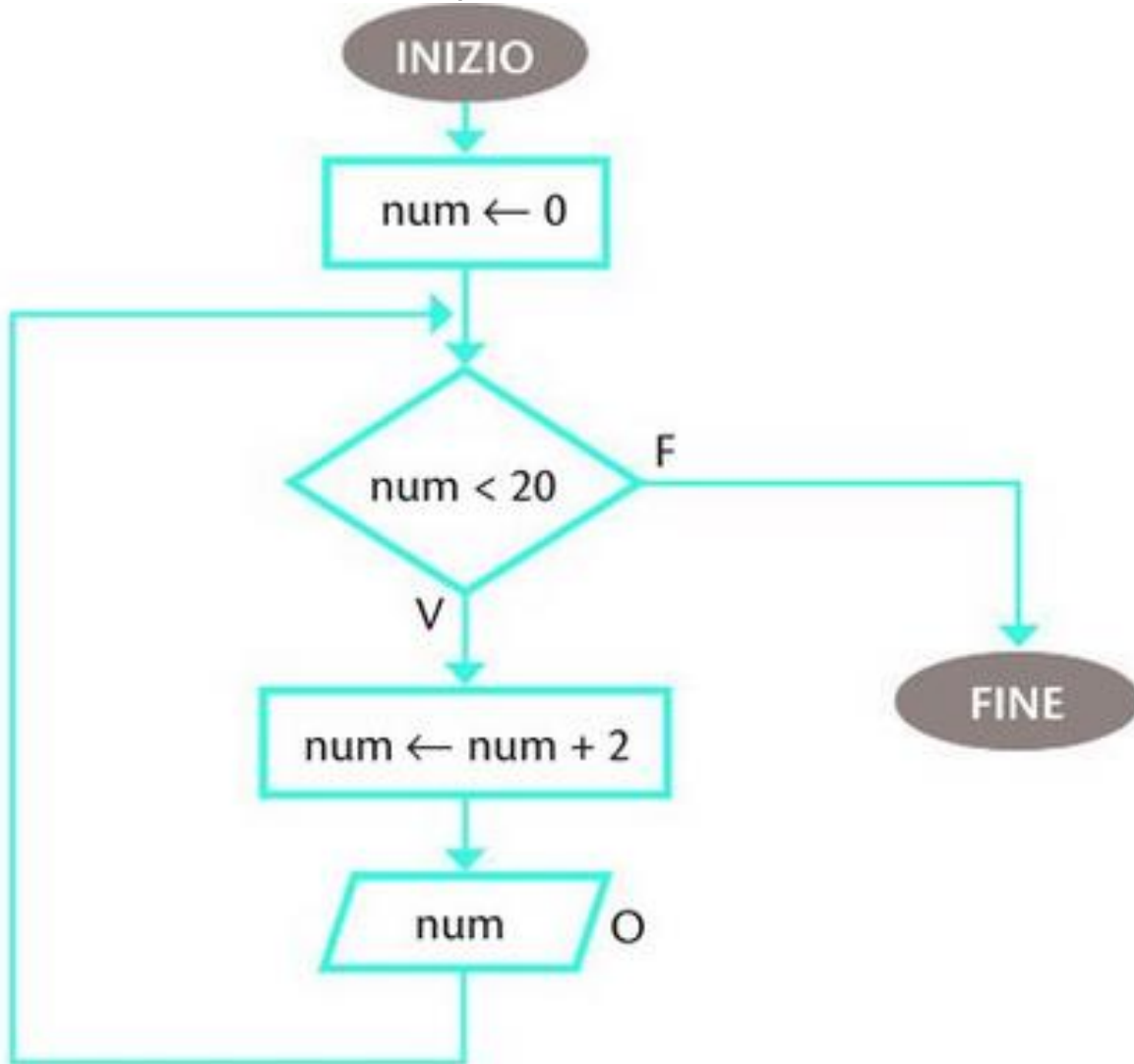
# Nei cicli pre/post e' necessario.....



È necessario, con entrambi i cicli, che l'esecuzione delle istruzioni che formano il corpo dell'iterazione modifichi il risultato della condizione, altrimenti il ciclo durerà all'infinito (**loop**).

# Esercizio: *iterazione pre-condizione*

Scrivere un programma che stampi in uscita una sequenza di numeri che rappresentano la tabellina del 2, a partire dal numero 2 sino al 20



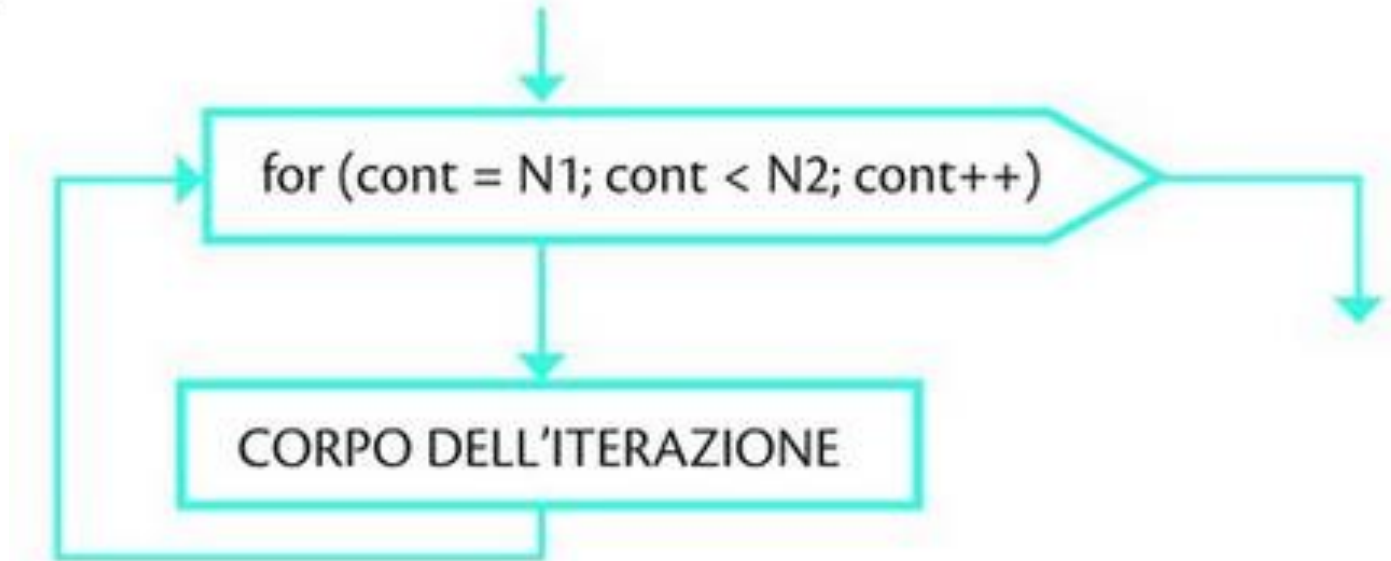
```
PROGRAMMA tabellina2
INIZIO
    num ← 0
    MENTRE num < 20 ESEGUI
        num ← num + 2
        SCRIVI (num)
FINE
```

```
int num;
num = 0;

//ELABORAZIONE
while (num<20);
{
    num = num + 2;
    System.out.print("Ecco il valore di num:"+num);
}
```

# IL CICLO *for*

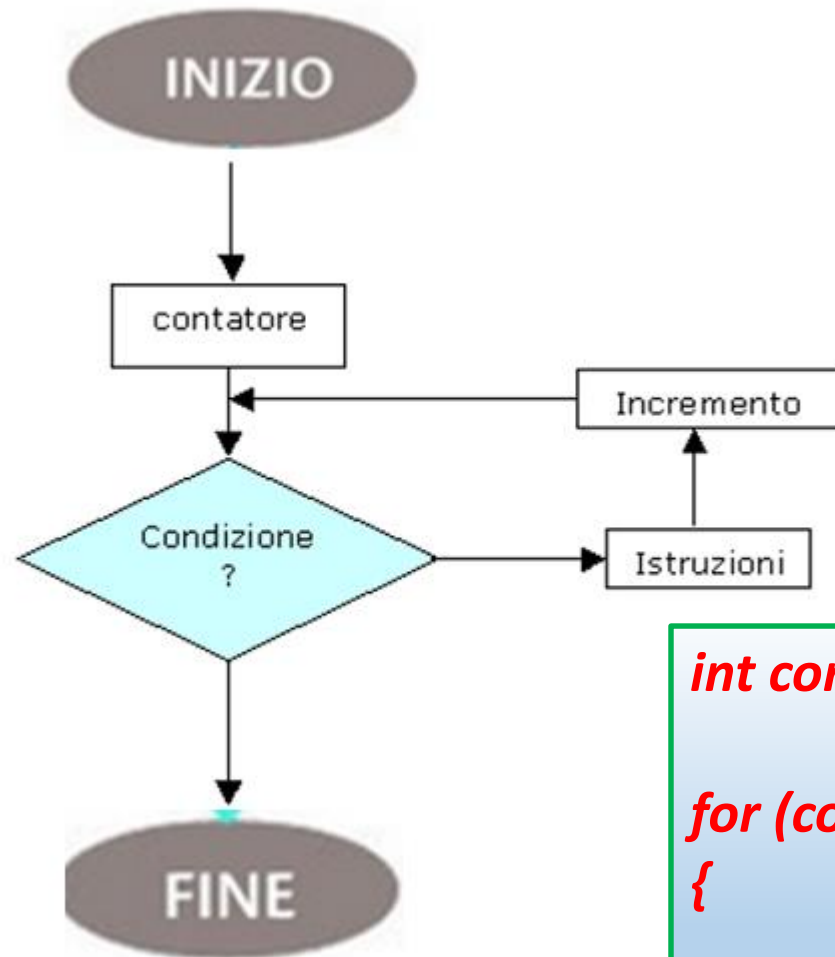
Quando si conosce il numero di volte che si desidera svolgere un ciclo, è possibile utilizzare la struttura **for** presente in molti linguaggi di programmazione.



Si tratta di un modo conciso per rappresentare un ciclo con controllo in testa, anche se spesso è considerato come una struttura autonoma. Anche le modalità di rappresentazione non sono univoche: qui ne mostreremo una. Il concetto è quello di incorporare in questa struttura la condizione di fine ciclo, la variabile di controllo del ciclo e l'incremento (o decremento) che questa deve avere a ogni iterazione

# Esercizio: *iterazione for*

Scrivere un programma che stampi in uscita una sequenza di numeri che rappresentano la tabellina del 2, a partire dal numero 2 sino al 20



```
PROGRAMMA tabellina2
INIZIO
    num ← 0
    MENTRE num < 20 ESEGUI
        num ← num + 2
        SCRIVI (num)
FINE
```

```
int contatore;
```

```
for (contatore=0; contatore<20; cotatore= contatore+incremento)
{
    System.out.print(+contatore);
}
```



**Esercizio: *for*:** *Scrivere un programma che stampi in uscita una sequenza di numeri che rappresentano la tabellina di un numero INTERO inserito da tastiera*

```
public class CicloPostCondizionaleFor {  
    public static void main(String[] args) {  
        int num, limite;  
        //istanzia un oggetto lettore di tipo Scanner  
        Scanner in = new Scanner(System.in);  
  
        System.out.print("Inserire il numero INTERO di  
                           cui si vuole conoscere la tabellina: ");  
        num = in.nextInt();  
        limite = num * 10;  
  
        for (num=0; num<=limite; num= num+2)  
        {  
            num = num + 2;  
            System.out.print("\t"+num);  
        } //for  
    } //main  
} //class
```

**output**

Inserire il numero INTERO di cui si vuole conoscere la tabellina: 2					
2	6	10	14	18	22

## FISSA LE CONOSCENZE

1. Quando è necessaria la struttura iterativa?
2. Che cosa costituisce il corpo dell'iterazione?
3. Che differenza c'è tra la struttura di iterazione con controllo in coda e quella con controllo in testa?

# Generazione di numeri casuali in JAVA

**Esercizio:** *Scrivere un programma che generi un numero casuale compreso tra 30 e 110*

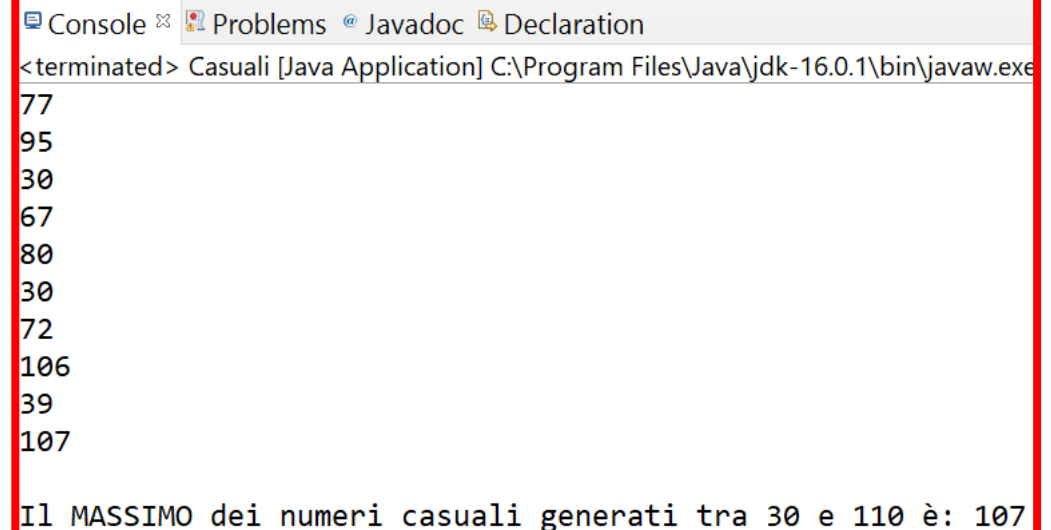
*Math.random()*

Il metodo *random()* della classe **Math** restituisce un numero casuale double compreso tra **0.0** e **1.0**. Calcolo l'intervallo  $[a,b] = 110-30 = 80$  e moltiplicando il numero restituito dalla funzione *random()* per 80 si ottiene un numero compreso tra **0.0** e **80.0**.

Utilizzando l'operazione di **casting**, si convertirà il numero casuale in un **intero** troncando la parte decimale, ottenendo un numero tra 0 e 80 Sommando **30** al numero ottenuto si ottiene un numero compreso tra **30** ed **110** (nel nostro caso  $a = 30$  e  $b = 110$ )

```
int numero = 30 + ((int) (80 * Math.random() ) )
```

```
public class Casuali {  
    public static void main(String[] args) {  
        int numero, massimo;  
        massimo = 0;  
  
        for(int i=0; i<10; i++)  
        {  
            //GENERO IL NUMERO CASUALE compreso tra 30 e 110  
            numero = 30 + (int) (80 * Math.random());  
            System.out.println(+numero);  
            if (numero>massimo) massimo= numero;  
        }  
  
        System.out.println("\nIl MASSIMO dei numeri casuali generati tra 30 e 110 è: "+massimo);  
    }  
}
```



```
<terminated> Casuali [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe  
77  
95  
30  
67  
80  
30  
72  
106  
39  
107  
  
Il MASSIMO dei numeri casuali generati tra 30 e 110 è: 107
```