Laboratorio di Informatica
Lezione 2

Cristian Consonni

30 settembre 2015

Outline

Chi sono

Cristian Consonni

- **DISI - Dipartimento di Ingegneria e Scienza dell'Informazione**
- **Pagina web** del laboratorio:
  http://disi.unitn.it/~consonni/teaching
- **Email**: cristian.consonni@unitn.it
- **Ufficio**: Povo 2 - Open Space 9
  - Per domande: scrivetemi una mail
  - Ricevimento: su appuntamento via mail

Outline for section 1

Commenti

- Commento su singola riga: doppio slash //

```
// Commento su singola riga
```

- Commento su singola riga: (apertura) slash + asterisco /* (chiusura) asterisco + slash */

```
/*
Commento multi riga.
Questo commento si può dipanare
su più di una riga
*/
```

Commenti (II)

Usate sempre commenti significativi:

- OK:

```
// inizializzo min a un valore che è sicuramente
// più piccolo dei valori assunti dal vettore
int min = -1;
```

- NO:

```
// inizializzo min a -1
int min = -1;
```

Stampa di stringe a schermo

- Concatenazione di stringhe:
  `System.out.println("Ciao, " + name + "!")`
- Print formatted:
  `System.out.printf("Ciao, %s!", name)`
  - `%d` per stampare *interi* (`int`);
  - `%f` per stampare *numeri con la virgola* (`float`, `double`);
  - `%s` per stampare *stringhe* (`String`);

Outline for section 2

Istruzione If-Else

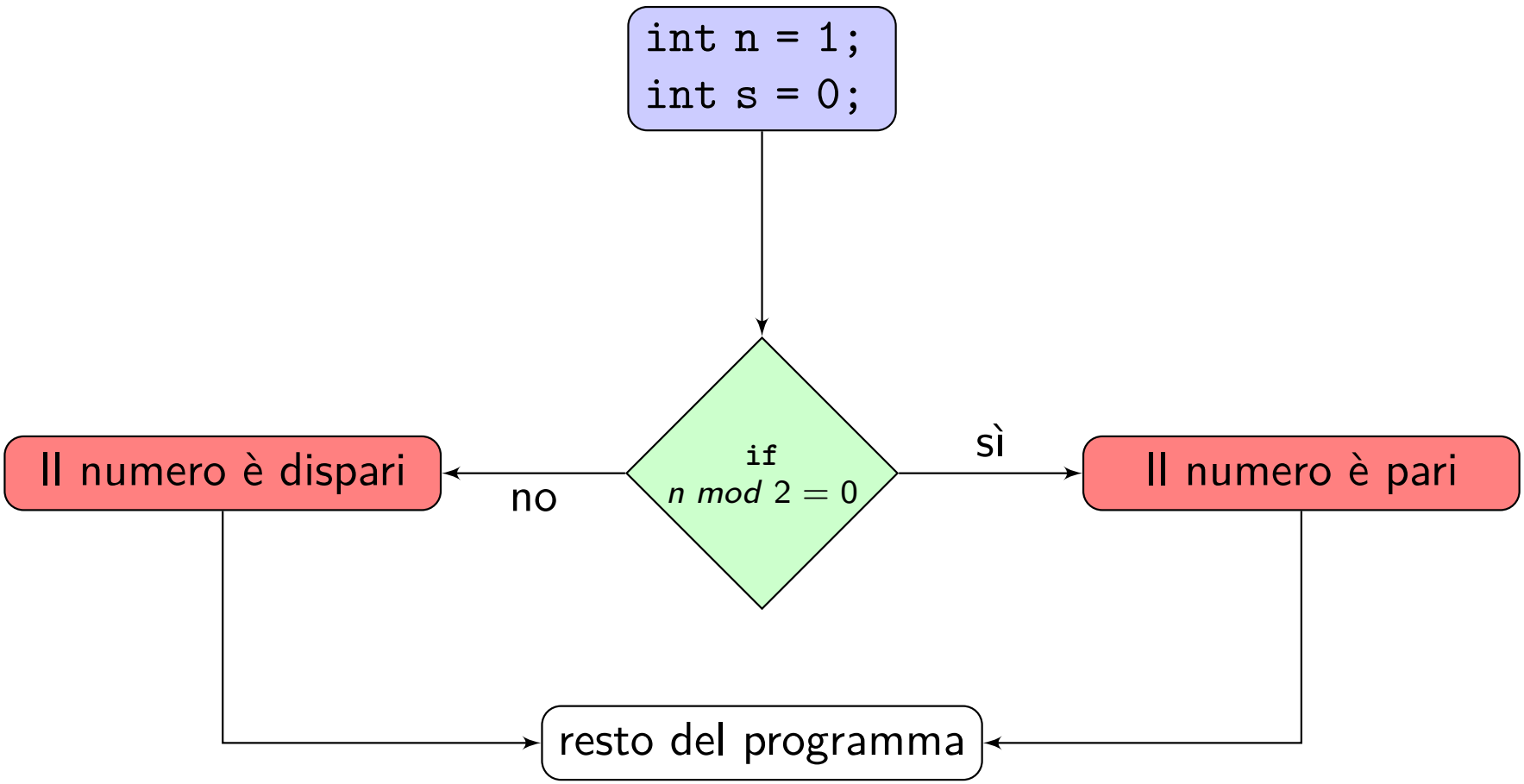
```
int n = 4;
if (n % 2 == 0) {
  System.out.println("Il numero è pari");
} else {
  System.out.println("Il numero è dispari");
}
```

Istruzione If-Else

```
int n = 4;
if (n % 2 == 0) {
  System.out.println("Il numero è pari");
} else {
  System.out.println("Il numero è dispari");
}
```

Istruzione If-Then-Else

Diagramma di flusso dell'istruzione If-Then-Else:



Outline for section 3

Iterazione (I)

Definzione di iterazione[1]:

> ≪*L'**iterazione**, chiamata anche **ciclo** o con il termine inglese **loop**, è una struttura di controllo [...] che ordina all'elaboratore di eseguire ripetutamente una sequenza di istruzioni, solitamente fino al verificarsi di particolari condizioni logiche specificate.*≫

Iterazione (II)

L'iterazione è utile per "tradurre" sommatorie e prodotturie:

- $$S(100) = \sum_{n=1}^{n=100} n$$

  calcola la somma di n **per** n **che va da** 1 **a** 100 (calcola la somma dei primi 100 interi)

- $$n! = \prod_{i=1}^{i=n} i$$

  calcola il prodotto di i **per** i **che va da** 1 **a** n (il fattoriale di n è il prodotto dei numeri da 1 a n.)

Iterazione (III)

L'iterazione traduce naturalmente formule ricorsive:

- Metodo della bisezione: metodo che trova numericamente gli zeri di una funzione continua;
- Metodo per il calcolo della radice quadrata di Newton. Si può calcolare la radice quadrata di un numero z nel modo seguente:

$$\begin{cases} x_0 = 0.5 \\ x_{n+1} = 0.5 \cdot x_n(3 - zx_n^2) \end{cases}$$

$$\lim_{n \to \infty} x_n = \sqrt{z}$$

Posso calcolare l'errore compiuto al termine n-esimo, x_n come $\varepsilon_n = |x_n^2 - z|$, Posso calcolare $\sqrt{z}$ a una data precisione: "prosegui a calcolare nuovi termini della successione x_n **finché l'errore non è più piccolo di** 10^{-3}".

Ciclo While (I)

Finché la condizione è **vera** esegui una certa operazione (ciclo `while-do` o semplicemente `while`):

Pseudocodice:

1: $s \leftarrow 0$
2: $n \leftarrow 1$
3: **while** $n \leq 100$ **do**
4: $\quad s \leftarrow s + n$
5: $\quad n \leftarrow n + 1$
6: **end while**

Java:

```
int s = 0;
int n = 1;
while (n <= 100) {
  s = s + n;
  n = n + 1;
}
```

Nel momento in cui la condizione all'interno del while diventa falsa si **esce** dal ciclo.

Ciclo While (II)

Finché la condizione è **vera** esegui una certa operazione (ciclo `while-do`):

Pseudocodice:

```
1: while condizione do
2:     comandi
3: end while
```

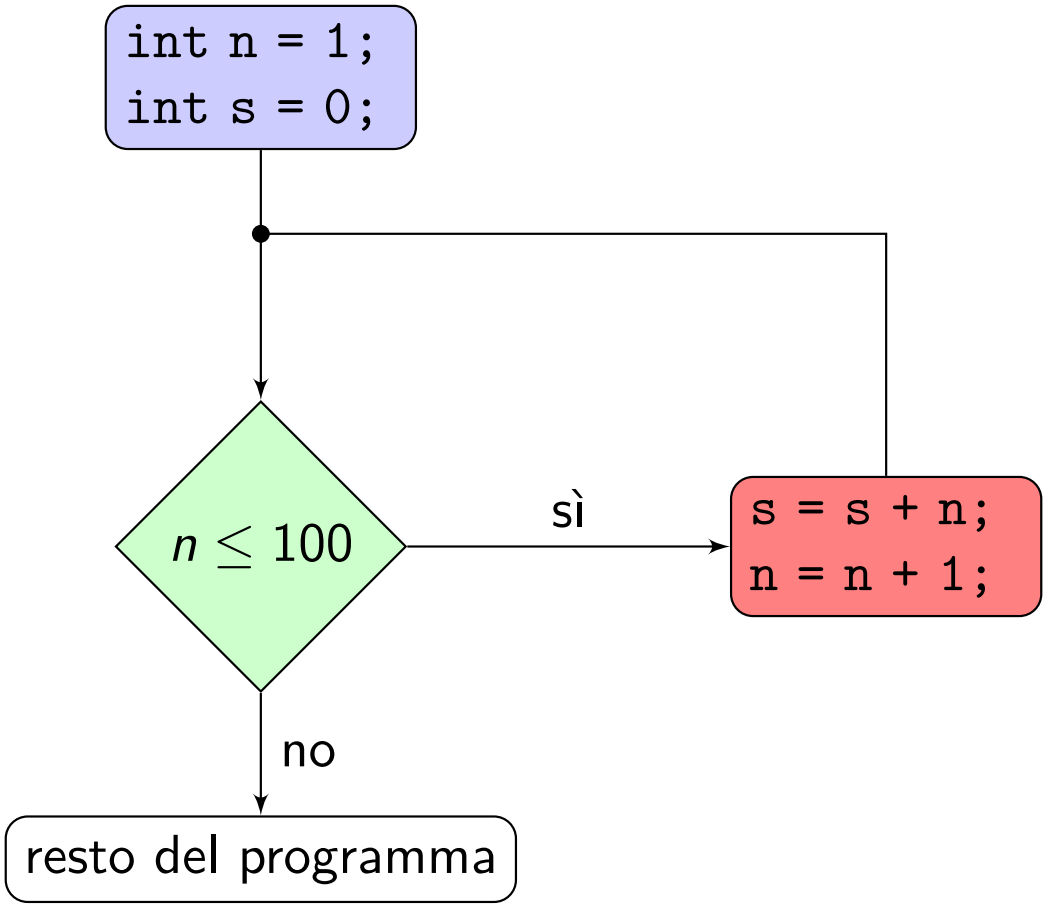
Java:

```
while (condizione) {
  comandi;
}
```

Nel momento in cui la condizione all'interno del while diventa falsa si **esce** dal ciclo.

Ciclo While (III)

Diagrama di flusso dell'istruzione While:



Ciclo While (IV)

Esecuzione del ciclo passo-passo:

step	n	s	$n \leq 100$**?**
–	1	0	–
1	1	0	sì
2	2	1	sì
3	3	3	sì
4	4	6	sì
...	...	...	...
99	99	4851	sì
100	100	4950	sì
101	101	4950	no
–	101	4950	

Ciclo While (VI)

Attenzione agli estremi dei cicli:

```
int n = 1;
while (n <= 10) {
  println("Ciao, mondo!");
  n = n + 1;
}
```

Scrive `Ciao, mondo!` **10** volte.

```
int n = 0;
while (n < 10) {
  println("Ciao, mondo!");
  n = n + 1;
}
```

Scrive `Ciao, mondo!` **10** volte.

Ciclo While (VI)

Se la condizione non diventa mai falsa, allora il ciclo **non termina mai**.

```
int s = 0;
int n = 99;
while (n != 0) {
  s = s + n;
  // n = 99, ..., 1,-1, ...
  n = n - 2;
}
```

```
int p = 0;
int n = 1;
while (p < 100) {
  p = p * n; // p == 0
  n = n + 1;
}
```

Ciclo While (VII)

Se la condizione è **sempre vera**, allora il ciclo non termina mai:

```
while (true) {
  s = s + n;
  n = n + 1;
}
```

Il comando **break** permette di uscire dal ciclo.

```
int s = 0;
int n = 1;
while (n <= 50) {
  // se n = 33 esco
  if (n == 33) {
    break;
  }
  s = s + n;
  n = n + 1;
}
```

Se devo saltare dei valori allora posso usare il comando **continue**:

$$s = \sum_{i=1, i \neq 3}^{50} i$$

```
int s = 0;
int n = 1;
while (n <= 50) {
  // salto il caso n = 3
  if (n == 3) {
    continue;
  }
  s = s + n;
  n = n + 1;
}
```

Nuovo progetto Eclipse

1. Aprire Eclipse
2. *File* $\rightarrow$ *New* $\rightarrow$ *Project* $\rightarrow$ *Java Project*
3. Inserire il nome e click su *Finish*
4. Click destro su *src* $\rightarrow$ *New* $\rightarrow$ *Class...*
5. Inserire il nome e check su `public static void main(String[] args)`
6. Click su *Finish*

Se avete difficoltà o problemi scrivetemi una mail!

Esercizi (I)

1. Scrivete un programma che stampi la stringa `Ciao, mondo!` a schermo per 10 volte;
2. Scrivere un programma che stampi tutti i numeri pari fino a 1000;

Ciclo Do-While (I)

Esegui una certa operazione **finché** la condizione è **vera** (ciclo `do-while`):

Pseudocodice:

1: $num \leftarrow 20$
2: $count \leftarrow 0$
3: **do**
4: $\quad num \leftarrow num/2$
5: $\quad count \leftarrow count + 1$
6: **while** $num \bmod 2 \neq 0$

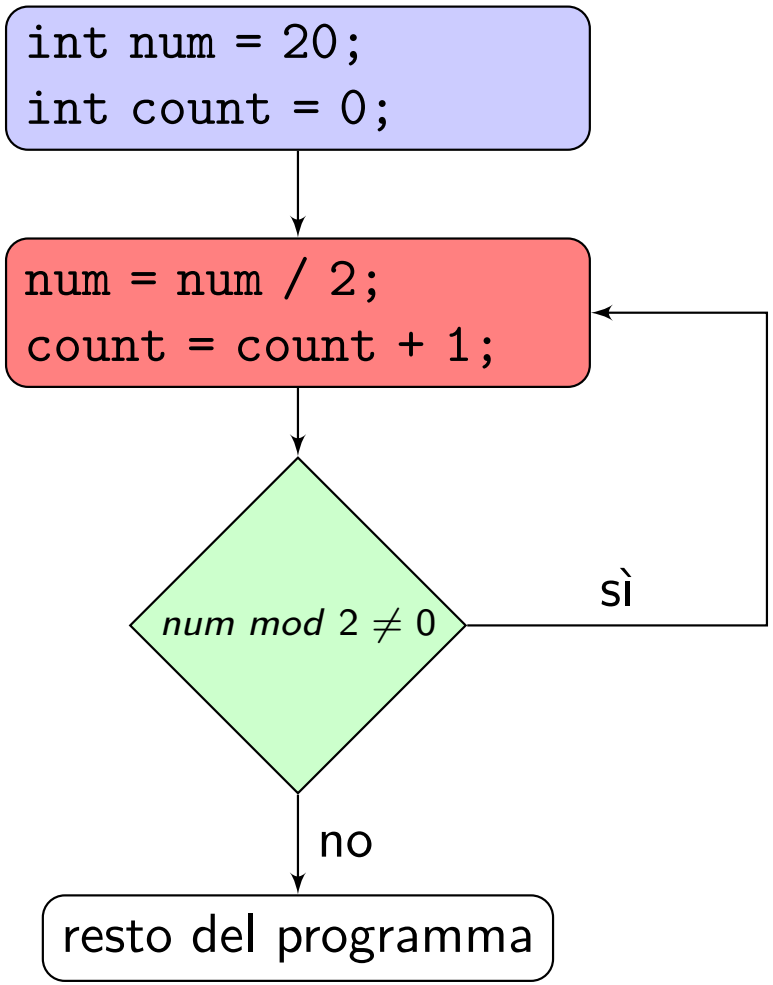
Java:

```
int num = 20;
int count = 0;
do {
  num = num / 2;
  count = count + 1;
} while (num % 2 != 0)
```

La differenza fondamentale tra la forma `while` e quella `do-while` è che con il `do-while` i comandi all'interno del ciclo vengono **sempre** eseguiti **almeno una volta**.

Ciclo Do-While (II)

Diagrama di flusso dell'istruzione Do-While:



Outline for section 4

Ciclo For (I)

Il ciclo `for` è un altro tipo di interazione dove si scorre una variabile, detta **indice** entro un intervallo di valori, con un dato **incremento**

```
for (inzializzazione; condizione incremento) {
  comandi;
}
```

Esempio:

```
int s = 0;
for (int n = 1; n <= 100; n++) {
  s = s + n;
}
```

Ciclo For (II)

Il ciclo `for` è un altro tipo di interazione dove si scorre una variabile, detta **indice** entro un intervallo di valori, con un dato **incremento**

```
for (inzializzazione; condizione; incremento) {
  comandi;
}
```

Note:

- nell'`inzializzazione;` potete **dichiarare** una variabile;
- Per l'`incremento;` potete usare step anche diversi da 1;

Ciclo For (III)

Quando la condizione è **falsa** il ciclo viene interrotto:

Pseudocodice:

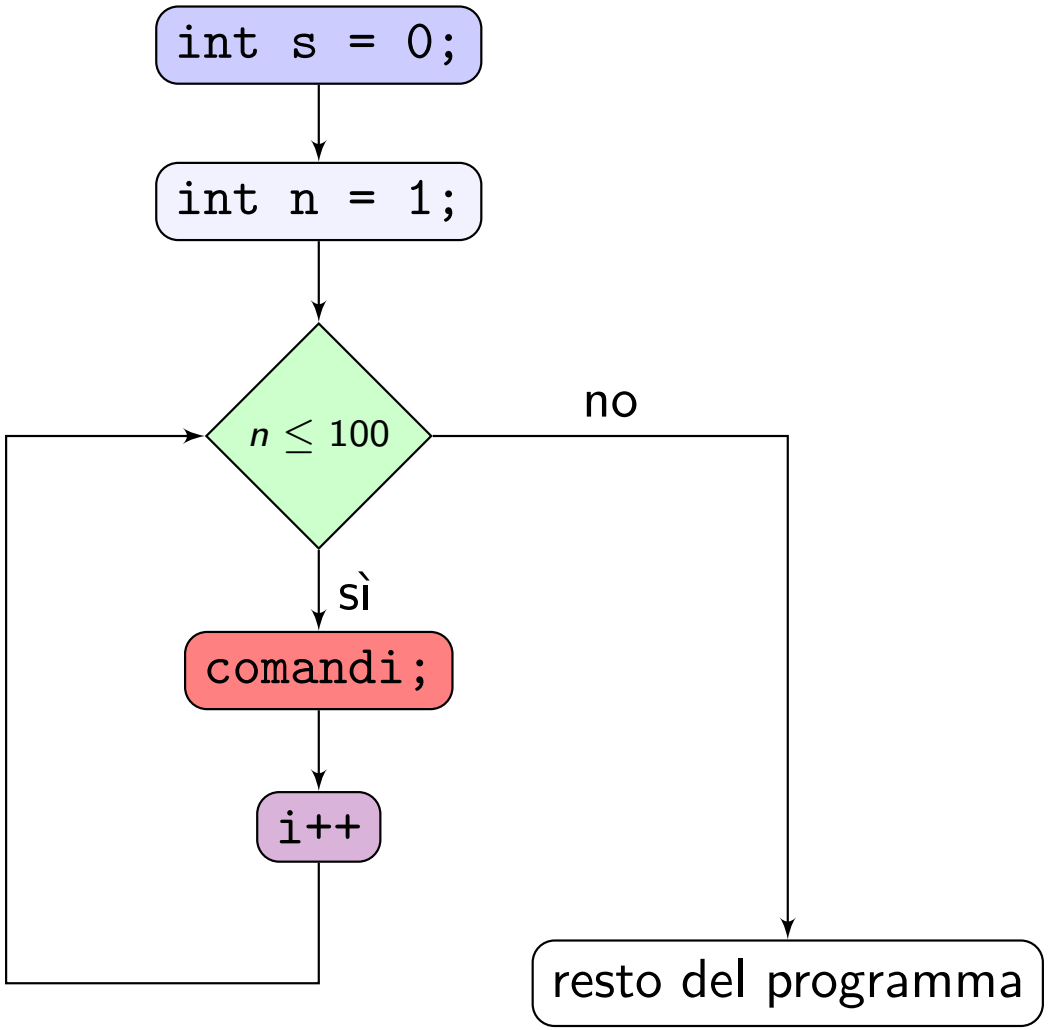
1: $s \leftarrow 0$
2: **for** $n \leftarrow 1$ to 100 by 1 **do**
3: $\quad s \leftarrow s + n$
4: **end for**

Java:

```
int s = 0;
for (int n = 1; n <= 100; n++) {
  s = s + n;
}
```

Ciclo For (IV)

Diagrama di flusso dell'istruzione for:



Esempi di ciclo `for`

- Nel ciclo i assume i valori $i = 0, 5, 10, 15, \ldots$

```
for (int i = 0; i < 100; i = i + 5) {
  comandi;
}
```

- Nel ciclo j assume i valori $j = 1, 2, 4, \ldots$

```
for (int j = 1; j < 1024; j = j * 2) {
  comandi;
}
```

- Nel ciclo k assume i valori $k = 10, 9, 8, \ldots$

```
for (int k = 10; k > 0; k--) {
  comandi;
}
```

Esistono gli operatori +=, -=, *=, ma vi consiglio di scrivere l'incremento in modo espicito per evitare errori.

Analogie tra ciclo `for` e ciclo `while`

I cicli **`for`** e **`while`** sono equivalenti. Tutti i cicli **`for`** possono essere "tradotti" in cicli **`while`** e **viceversa**.

```
for (inzializzazione; condizione; incremento) {
  comandi;
}
```

```
inzializzazione;
for (condizione) {
  comandi;
  incremento;
}
```

Outline for section 5

Esercizi (I)

- Scrivete un programma che stampi la canzone popolare inglese "*99 bottiglie di birra*"
- (vedete anche `https://esolangs.org/wiki/99_bottles_of_beer`)

  ≪99 bottles of beer on the wall, 99 bottles of beer.
  Take one down, pass it around, 98 bottles of beer on the wall
  99 bottles of beer on the wall, 99 bottles of beer.
  Take one down, pass it around, 98 bottles of beer on the wall
  ...
  1 bottle of beer on the wall, 1 bottle of beer.
  Take one down, pass it around, no more bottles of beer on the wall
  There no more bottles of beer on the wall, no more bottles of beer.≫

Esercizi (II)

- Utilizzando il ciclo `while` scrivete un programma che dato un intero stampi a schermo la "tabellina". Ad esempio, se il numero è 7 dovrete stampare a schermo:
  - 7*0 = 0
  - 7*1 = 7
  - 7*2 = 14
  - ...
  - 7*10 = 70
- riscrivete il programma precendente usando il ciclo `for`.

Esercizi (III)

- Scrivete un programma che calcoli il fattoriale di un numero intero a vostra scelta.

La definizione del fattoriale è la seguente:

$$n! = n \times (n-1) \times \cdots \times 1 \tag{1}$$

quindi il calcolo del fattoriale può essere definito da:

```
1: fatt ←?                    ▷ Quale valore va messo qui?
2: for i ← 1 to N do
3:     fatt ← fatt × i
4: end for
```

Esercizi (IV)

- Scrivere un programma che stampi i valori della serie di Fibonacci minori di 10000. La serie di Fibonacci è definita da:

$$\begin{cases} x_0 = 1 \\ x_1 = 1 \\ x_{n+1} = x_n + x_{n-1} \end{cases}$$

Esercizi (V)

- Scrivere un programma che usi il metodo per il calcolo della radice quadrata di Newton.

$$\begin{cases} x_0 = 0.5 \\ x_{n+1} = 0.5 \cdot x_n(3 - zx_n^2) \end{cases}$$

$$\lim_{n \to \infty} x_n = \sqrt{z}$$

Il programma deve calcolare la serie definita sopra fino a che l'errore $\varepsilon_n = |x_n^2 - z|$, non è più piccolo di 10^{-3}. Per il valore assoluto utilizzate la funzione `Math.abs()`.

Esercizi (VI)

- Metodo della bisezione

https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/10RootFinding/bisection/bisection.gif

Input: Function f, endpoint values a, b, tolerance ε, maximum iterations N_{MAX} $a < b$, either $f(a) < 0$ and $f(b) > 0$ or $f(a) > 0$ and $f(b) < 0$
Output: value which differs from a root of $f(x) = 0$ by less than ε

```
1: N ← 1
2: while N ≤ N_MAX do
3:     c ← (a + b)/2
4:     if f(c) = 0 ∨ (b–a)/2 < ε then
5:         print(c)
6:         return ;
7:     end if
8:     N ← N + 1
9:     if sign(f(c)) = sign(f(a)) then
10:        a ← c
11:    else
12:        b ← c
13:    end if
14: end while
15: print(Non ho trovato risultati)
```

Esercizi (VII)

Test di primalità

- Scrivere un programma che, dato un intero positivo, verifichi se quel numero è primo oppure no.

Un numero $n \in \mathbb{N}, n > 1$ è **primo** se e solo se è divisibile solo per 1 e per se stesso.