Laboratorio di Pensiero Computazionale

Simone Zennaro – simone.zennaro@unipd.it

Nella lezione di oggi

- Cos'è un linguaggio di programmazione e perché è necessario
- Flow Chart
- Lo Pseudo-codice
- Espressioni condizionali e cicli
- Esercizi

Linguaggio di programmazione

Il linguaggio di programmazione è una notazione non ambigua con cui si descrive un procedimento computazionale.

Permette di esprimere l'insieme dei passi che una macchina compie per risolvere un problema (algoritmo)

Sintassi: l'insieme di regole formali per la scrittura di programmi in un linguaggio, che dettano le modalità per costruire frasi corrette nel linguaggio stesso

Semantica: l'insieme dei significati da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio

NB: una frase può essere sintatticamente corretta e tuttavia non avere significato!

Compilati VS interpretati

- **Compilati** i linguaggi che necessitano di un programma chiamato compilatore per convertire il sorgente in codice macchina. Il processo di compilazione è strettamente connesso con la piattaforma hardware/software su cui è stato scritto il codice. Tali software necessitano quindi di una completa ricompilazione, ed una parziale riscrittura, per poter essere eseguiti su un sistema diverso da
- **Interpretati** Questa famiglia di linguaggi non trasforma direttamente le proprie istruzioni in codice macchina ma appalta il tutto ad un software interprete. Tali applicativi sono nativamente “multipiattaforma” e non devono essere adattati ogni volta per essere usati su più ambienti software. Gli sviluppatori quindi non sono obbligati ad eseguire un lavoro di porting, basta appunto rendere disponibile il software interprete su più sistemi operativi.

Gerarchia di astrazione

6 Programma applicativo
5 Linguaggio di programmazione
4 Linguaggio assembly (assemblativo)
3 Nucleo di sistema operativo
2 Linguaggio macchina
1 Microprogramma
0 Logica digitale

Problem Solving: Flow Chart

https://www.drawio.com/doc/getting-started-basic-flow-chart

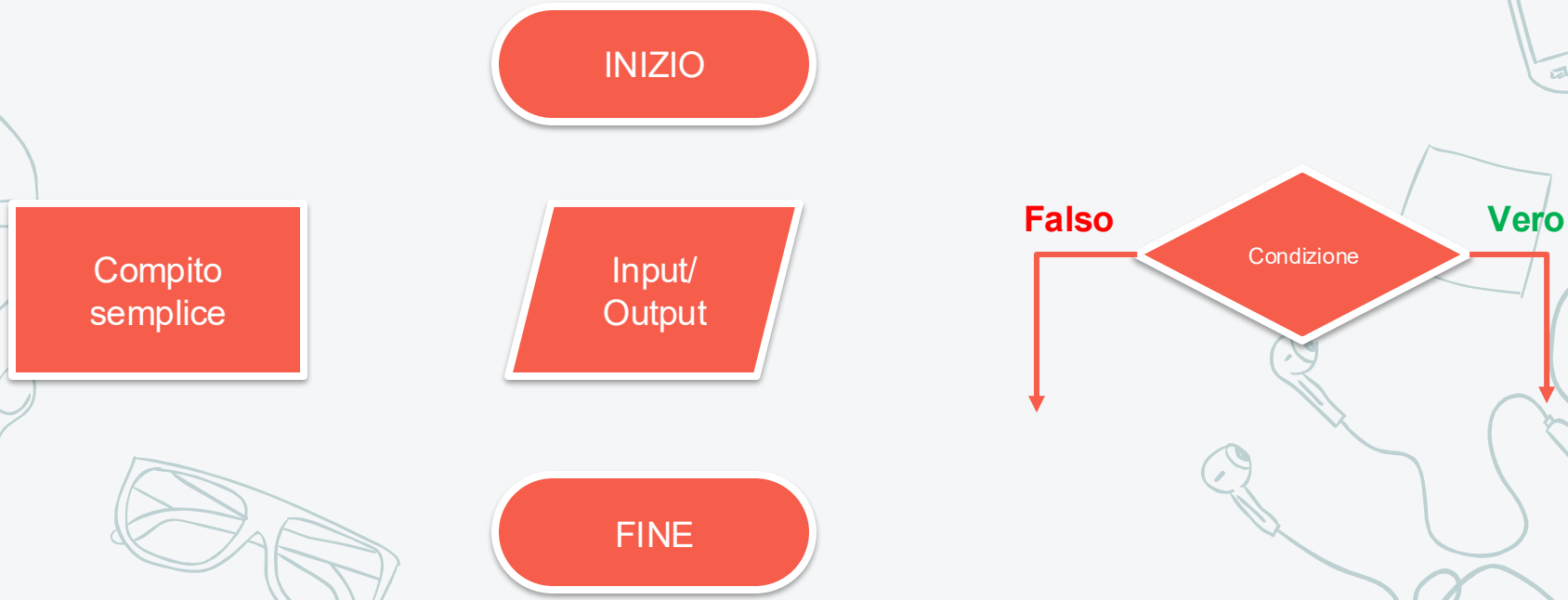
I diagrammi di flusso (flowcharts) sono strumenti visuali per rappresentare algoritmi/procedure in modo chiaro e lineare.
Alcuni vantaggi:

- Permettono di schematizzare la logica del programma prima di scrivere codice, aiutando a scoprire errori logici.
- Aiutano nella comunicazione: anche persone non esperte di codice possono capire il flusso di esecuzione.
- Utile per documentazione, debugging, pianificazione.
- Possono mostrare decisioni (if/else), cicli (while, for), input/output, processamento, ecc.

Problem Solving: Flow Chart

https://www.drawio.com/doc/getting-started-basic-flow-chart

- Un diagramma di flusso (flow chart) mostra la struttura delle decisioni e delle attività necessari a risolvere un problema
- Elementi principali dei flow chart:

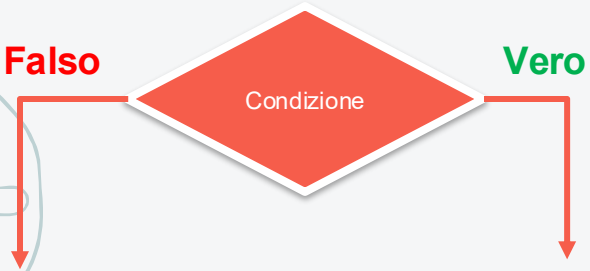


Problem Solving: Flow Chart

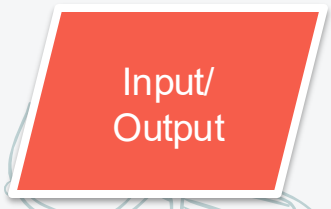
https://www.drawio.com/doc/getting-started-basic-flow-chart

Inizio / fine del processo

Un'operazione, elaborazione dati



Punto in cui si prende una decisione
-> biforcazione logica



Input o output di dati

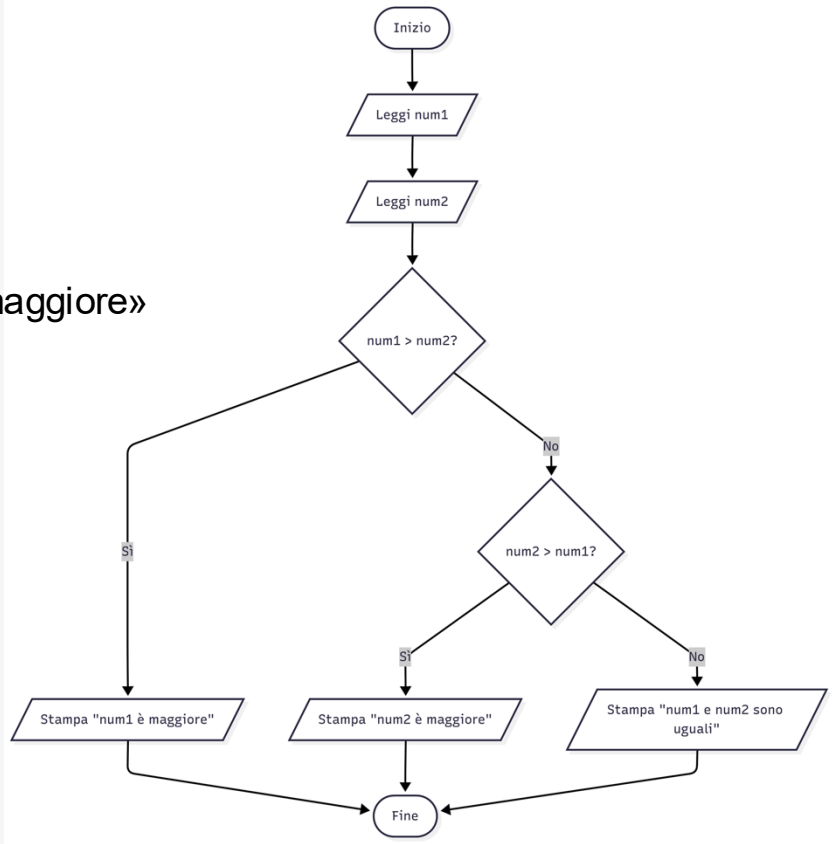
Esempio

Supponiamo di voler scrivere un semplice programma che:

- ✕ Chiede all'utente due numeri
- ✕ Stampa quale dei due è maggiore
- ✕ Se sono uguali, lo indica

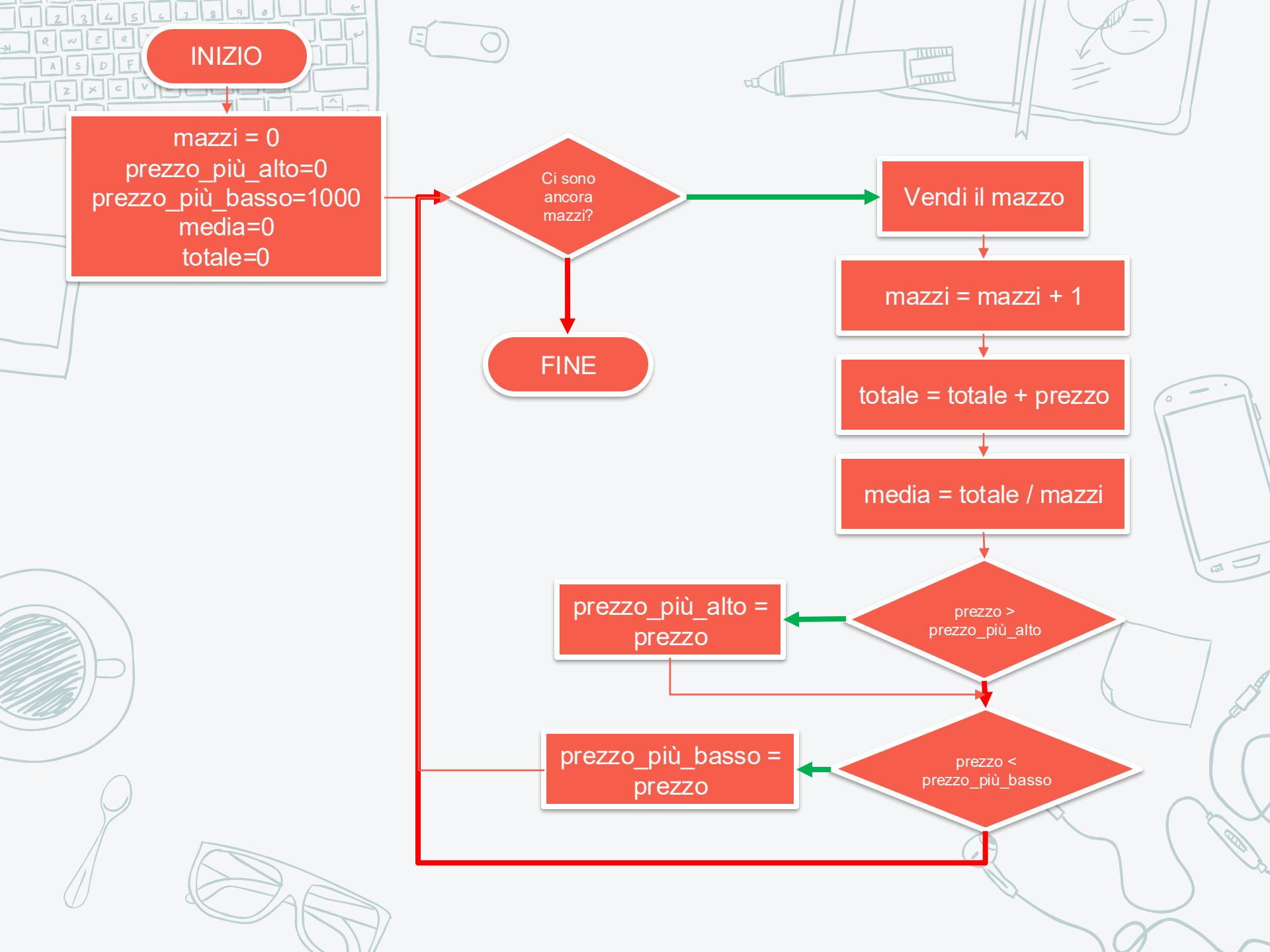
Esempio

1. Inizio
2. Leggi num1
3. Leggi num2
4. Se num1 > num2 ➔ stampa » num1 è maggiore»
   Altrimenti se num2 > num1 ➔ stampa » num2 è maggiore»
   Altrimenti ➔ stampa » Sono uguali »
5. Fine



Esercizio – disegnare il diagramma di flusso

- Fred vende mazzi di fiori al centro commerciale
- Un giorno Joe, il capo di Fred, gli dice che in qualunque momento della giornata, ha bisogno di sapere:
  - Quanti mazzi di fiori sono stati venduti
  - Qual era il valore del mazzo più costoso venduto
  - Qual era il valore del mazzo meno costoso venduto
  - Qual era il valore medio dei mazzi venduti


INIZIO
mazzi = 0
prezzo_più_alto=0
prezzo_più_basso=1000
media=0
totale=0
Ci sono ancora mazzi?
FINE
Vendi il mazzo
mazzi = mazzi + 1
totale = totale + prezzo
media = totale / mazzi
prezzo > prezzo_più_alto
prezzo_più_alto = prezzo
prezzo < prezzo_più_basso
prezzo_più_basso = prezzo

Problem Solving: pseudocodice

Lo pseudocodice è un modo per descrivere un algoritmo **in linguaggio naturale semplificato**, senza usare la sintassi di un linguaggio di programmazione.

Serve per:

- ✕ la logica di un programma in modo chiaro;
- ✕ essere indipendente da un linguaggio specifico (Python, C, Java...);
- ✕ trasformarsi facilmente in codice vero e proprio.

Pseudocodice

```
1  Scrivi: "Inserisci la tua età: "
2  Se l'età è uguale o maggiore di 18:
3      Scrivi "Sei maggiorenne"
4  Altrimenti:
5      Scrivi "Non sei maggiorenne"
```

Python

```
1  age=input("Inserisci la tua età: "))
2  if int(age)>=18:
3      print("Sei maggiorenne")
4  else:
5      print("Non sei maggiorenne")
```

Esempio di problem solving

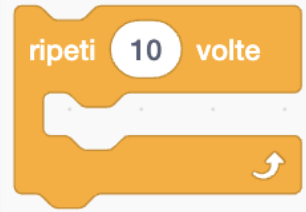
- Problema: Calcola il massimo tra A e B
- Soluzione: Il massimo è il numero maggiore tra A e B
- Soluzione Formale:

```
1.  max = 0
2.  se A > B allora:
            max = A
3.  altrimenti:
            max = B
4.  stampa max
```

I BLOCCHI DI SCRATCH

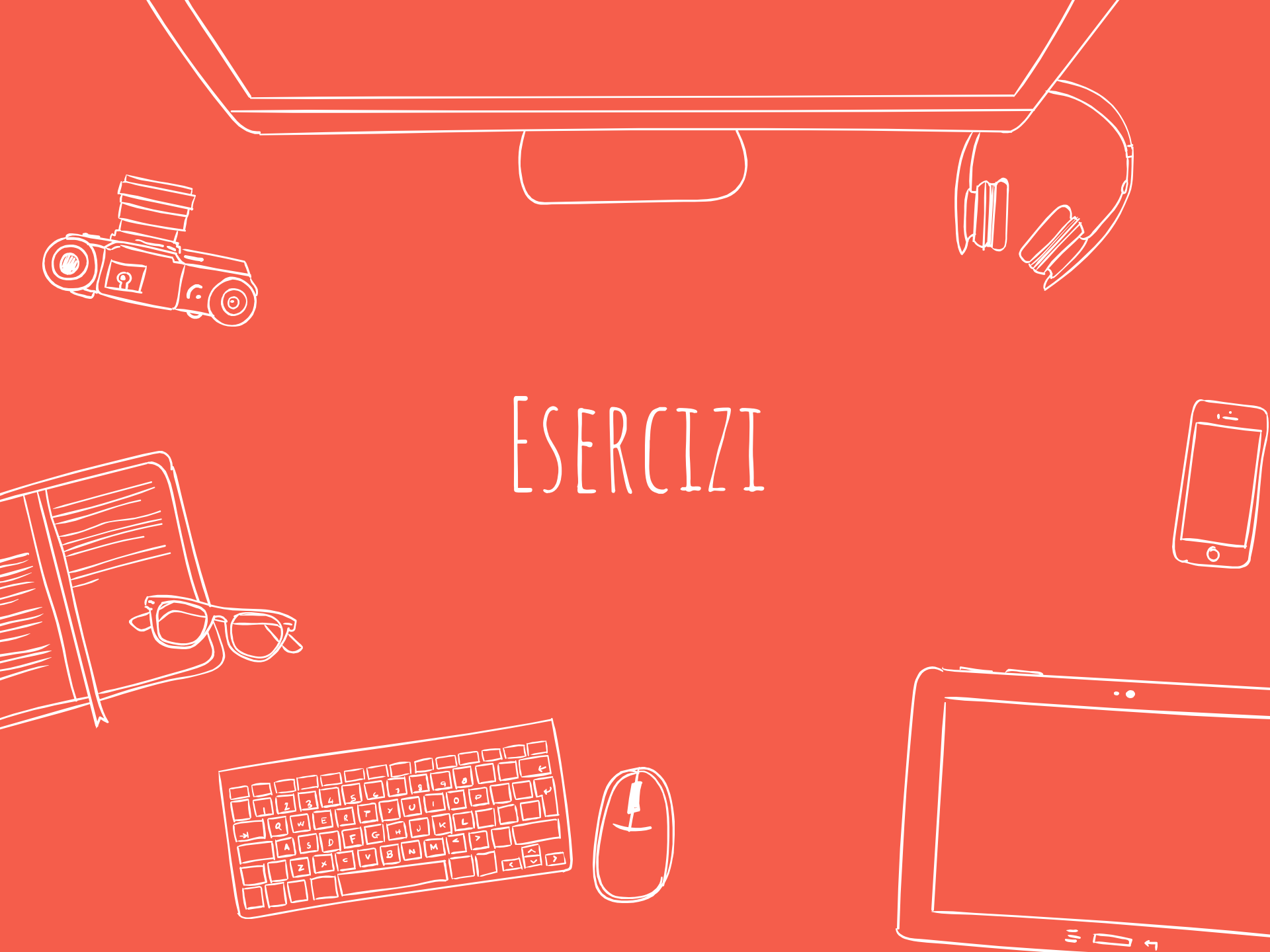
- Possiamo esercitarci anche usando i blocchi in Scratch

ESPRESSIONE CONDIZIONALE



CICLI

Esercizi



Esercizio 1

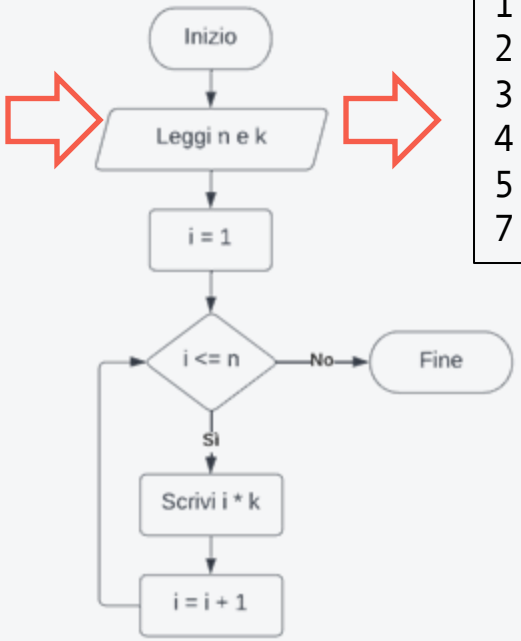
Dati n,k > 0, calcolare i primi n multipli di k.

1. Scrivere la procedura risolutiva per punti
2. Disegnare diagramma di flusso
3. Scrivere pseudocodice

Esercizio 1

- Dati n,k > 0, calcolare i primi n multipli di k.
- La soluzione del problema è piuttosto elementare: letti in input i dati del problema, si iterano le istruzioni con cui si calcola l'i-esimo multiplo di k (per i = 1,2,…,n) come prodotto i × k e si stampa il risultato.

```
1: leggi n e k
2: i=1
3: se i <= n prosegui,
altrimenti vai al passo 6
3: scrivi i×k
4: i=i+1
5: vai al passo 3
6: stop
```




```
1: leggi n e k
2: i=1
3: fintanto che i <= n
4:       scrivi i * k
5:       i = i + 1
7: stop
```

https://think.cs.vt.edu/blockpy/

Esercizio 2

Dati n,k > 0 stabilire se n è divisibile per k

1. Scrivere la procedura risolutiva per punti
2. Disegnare diagramma di flusso
3. Scrivere pseudocodice