



UNIVERSITÀ  
DEGLI STUDI DI TRIESTE



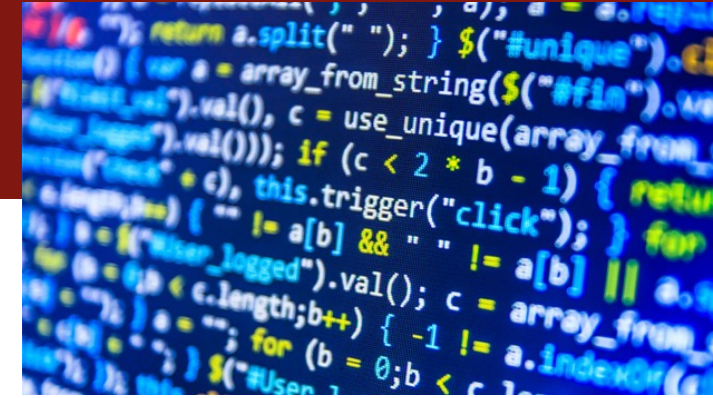
**Corso di Laurea in Tecniche di Radiologia Medica per immagini e Radioterapia  
Sistemi Elettronici e informatici in ambito di Imaging I**

**1CFU – 10 ore**

**CENNI DI PROGRAMMAZIONE**

***Prof. Sara Renata Francesca Marceglia***

# Software



Users



Hardware

Interfaccia utente/macchina



## Application Software (programmi)

**Programma = Algoritmo** scritto in modo da poter essere eseguito da un calcolatore (esecutore automatico)



Per scrivere un programma è necessario rappresentare istruzioni e dati in un formato tale che l'esecutore automatico sia capace di memorizzare e manipolare.

# Algoritmi

- **Definizione**

- Il nome deriva da Abū Jaʿfar Muḥammad ibn Mūsā al-Khwārizmī
- Insieme finito di istruzioni che permettono di RISOLVERE UN PROBLEMA



Tutti i programmi sono algoritmi, ma non tutti gli algoritmi sono programmi

- **In programmazione:**

- Descrizione della soluzione di problema scritta in modo da poter essere eseguita da un esecutore (eventualmente diverso dall'autore dell'algoritmo)
- Sequenza di istruzioni che operano su dati.

## Esempio

Mettete in una **ciotola l'uovo intero** (o due piccole) insieme allo **zucchero** e sbatteteli con una forchetta oppure con una frusta a mano.

Aggiungete il **latte** a temperatura ambiente e continuate a sbattere solo **con la forchetta o con una frusta a mano** l'impasto per i pancake senza burro.

Aggiungete a poco a poco tutta la **farina setacciata e il lievito in polvere** per dolci e continuate a mescolare con una frusta a mano in modo che **non si formino grumi** fino a che otterrete una pastella liscia e omogenea.

(se si dovessero formare grumi potete usare lo sbattitore per qualche secondo).

Fate scaldare **una padella antiaderente** e, quando sarà ben calda, mettete a cuocere un mestolo di impasto dei pancake senza burro e fate cuocere **per circa 4 minuti** fino a che vedrete che si formeranno delle bollicine sopra quindi girate i pancake e fateli cuocere dall'altra parte per altri **3 o 4 minuti**.



# Esempio

**14 min** (5,5 km)

tramite Via di Colonia

Traffico moderato come al solito



## via Valmaura 13 (stadio Nereo Rocco)

34148 Trieste TS

- Prendi Via dell'Istria e Galleria di Montebello in direzione di Via Pier Paolo Vergerio

4 min (2,0 km)

- Prendi Via dei Piccardi e Via Antonio Canova in direzione di Via Cesare Battisti

6 min (1,8 km)

- Prendi Via Giulia in direzione di Via di Colonia

1 min (350 m)

- 📍 Alla rotonda prendi la 2ª uscita e prendi Via di Colonia

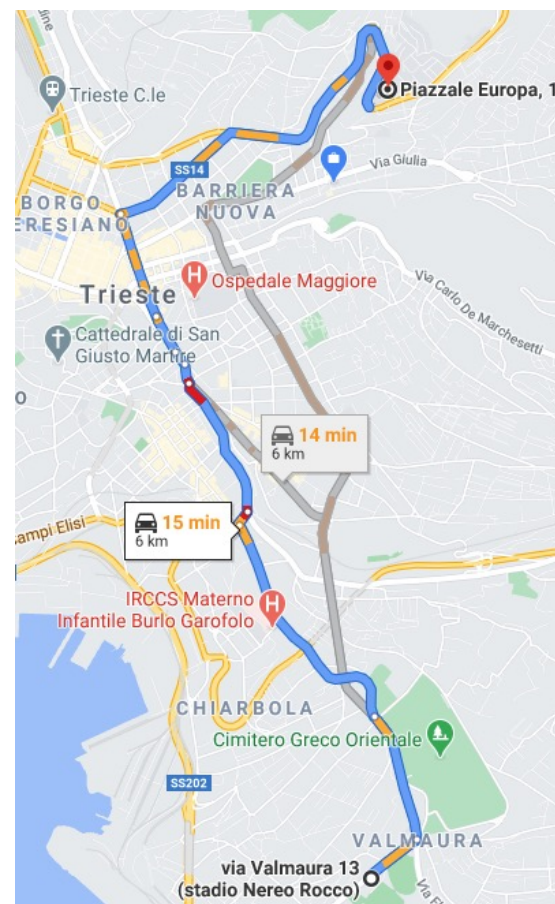
2 min (1,0 km)

- Svolta a destra e prendi Via Fabio Severo/SS14

40 s (400 m)

## Piazzale Europa, 1

34127 Trieste TS



**Esempio: seguire la lezione di «sistemi di elaborazione...» su Teams**

# Il percorso di programmazione

## Identificazione del problema

- Problema: classe di domande omogenee (concetto astratto)
- Richiesta: può essere l'istanza di un problema (caso specifico)
- Esempio: sommare i numeri 2 e 3 è l'istanza del problema generale di somma di due numeri ( $A+B$ )
- Per trovare la soluzione: dall'istanza devo risalire al problema e risolverlo a livello astratto → poi funzionerà per tutti i problemi di quella classe

## Analisi del problema

- Comprensione di dati INPUT e dati OUTPUT (cosa mi è fornito, cosa mi è richiesto)

## Risoluzione del problema

- Definizione di una trasformazione  $F$  che dati i dati in input mi fornisca i dati in output

## Verifica della soluzione

- La soluzione deve essere testata
- Va definito il modello di test



## Esempio: istanza, problema e soluzione

### **ISTANZA DEL PROBLEMA**

**VOGLIO ORDINARE IN MODO CRESCENTE IL VETTORE**

**[2 77 1 935 11 19 773 15 3]**

### **IDENTIFICAZIONE DEL PROBLEMA**

**ORDINAMENTO DEI DATI IN UN VETTORE**

### **ANALISI DEL PROBLEMA**

**INPUT: VETTORE**

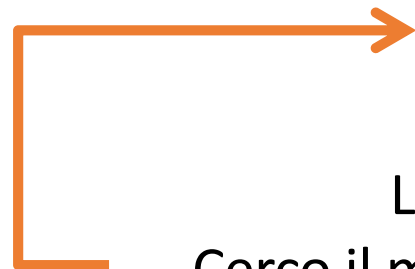
**OUTPUT: VETTORE ORDINATO**

## Esempio: istanza, problema e soluzione

**DATO IL VETTORE IN INGRESSO**

**[2 77 1 935 11 19 773 15 3]**

Modo 1: ho a disposizione  
la funzione "minimo"

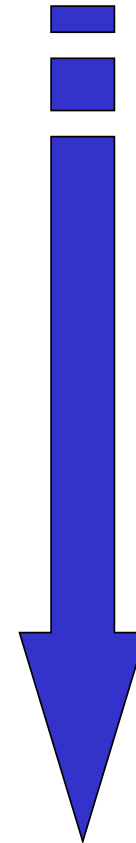


Cerco il minimo  
Lo metto da parte  
Lo elimino dal vettore  
Cerco il minimo nel vettore rimanente



**VETTORE IN USCITA**

**[1 2 3 11 15 19 77 773 935]**



## Esempio: istanza, problema e soluzione

Modo 2: non ho a disposizione la funzione "minimo", ma posso solo confrontare due numeri

**DATO IL VETTORE IN INGRESSO**

**[2 77 1 935 11 19 773 15 3]**



Scompongo il problema in sottoproblemi usando l'unica operazione che posso fare: riordinare vettori di due numeri (vedo quale è maggiore e lo metto per secondo) →

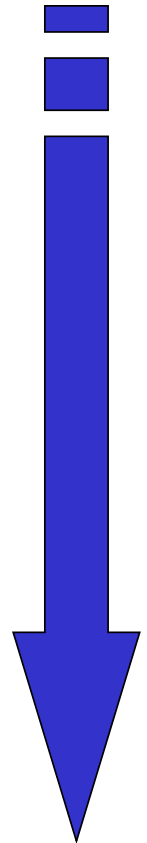
- 1- vettore composto da 3 numeri: ripeto l'ordinamento del vettore da due numeri per 3 volte (3 vs 2 e 2 vs 1)
- 2- vettore di 4 numeri: ripeto l'ordinamento 3 volte (4 vs 3, 3 vs 2 e 2 vs 1)

3 – proseguo finchè non arrivo alla lunghezza finale



**VETTORE IN USCITA**

**[1 2 3 11 15 19 77 773 935]**



## Esempio: istanza, problema e soluzione

[2 77 1 935 11 19 773 15 3]

[2 77 1 935 11 19 773 15 3] → [2 77 1 935 11 19 773 15 3]

[2 77 1 935 11 19 773 15 3] → [2 1 77 935 11 19 773 15 3] →  
[1 2 77 935 11 19 773 15 3]

[1 2 77 935 11 19 773 15 3] → [1 2 77 935 11 19 773 15 3] →  
[1 2 77 935 11 19 773 15 3] → [1 2 77 935 11 19 773 15 3]

[1 2 77 935 11 19 773 15 3] → [1 2 77 11 935 19 773 15 3] →  
[1 2 11 77 935 19 773 15 3] → [1 2 11 77 935 19 773 15 3] →  
[1 2 11 77 935 19 773 15 3]

...  [1 2 3 11 15 19 77 773 935]

## La formalizzazione della soluzione

### LINGUAGGIO DI RAPPRESENTAZIONE



- Soluzione astratta del problema
- Sequenza ordinata di azioni elementari

- Sequenza di azioni comprensibili all'esecutore
- Soluzione adattata all'esecutore (CPU)

## Esempio: algoritmo e linguaggio

**Problema: Calcolare la data successiva ad una data fornita in input**

Data in input

**Pseudo  
linguaggio**

Se il giorno non è l'ultimo del mese → aggiungo un giorno e mantengo inalterato mese e anno

Se il giorno è l'ultimo del mese e il mese non è dicembre → output è il primo giorno del mese successivo

Se il mese è dicembre → il giorno successivo è il 1 gennaio dell'anno successivo

## Esempio: algoritmo e linguaggio

### Problema: Calcolare la data successiva ad una data fornita in input

```
int main()
{
    int giorno,mese,anno;
    puts("Dammi 3 numeri, rispettivamente giorno, mese ed anno:");
    scanf("%d%d%d",&giorno,&mese,&anno);
    if (giorno<30)
        giorno=giorno+1;
    else {
        if (giorno==30 && mese==12){
            giorno=1;
            mese=1;
            anno=anno+1;
        }else {
            if (giorno==30 && mese<12)
                giorno=1;
            mese=mese+1;
            anno=anno;
        }
    }
    printf("%d/%d/%d",giorno,mese,anno);
    system("PAUSE");
    return 0;
}
```

Linguaggio C

## Gestione dei dati: Variabili, Costanti, Array

- Nei programmi abbiamo bisogno di dati di input
- Solitamente I programmi operano con dati che posso essere differenti nelle diverse applicazioni (ad esempio la data da cui partire per calcolare la successiva)

**VARIABILE =  
locazione di memoria che può essere usata per  
memorizzare un valore.**

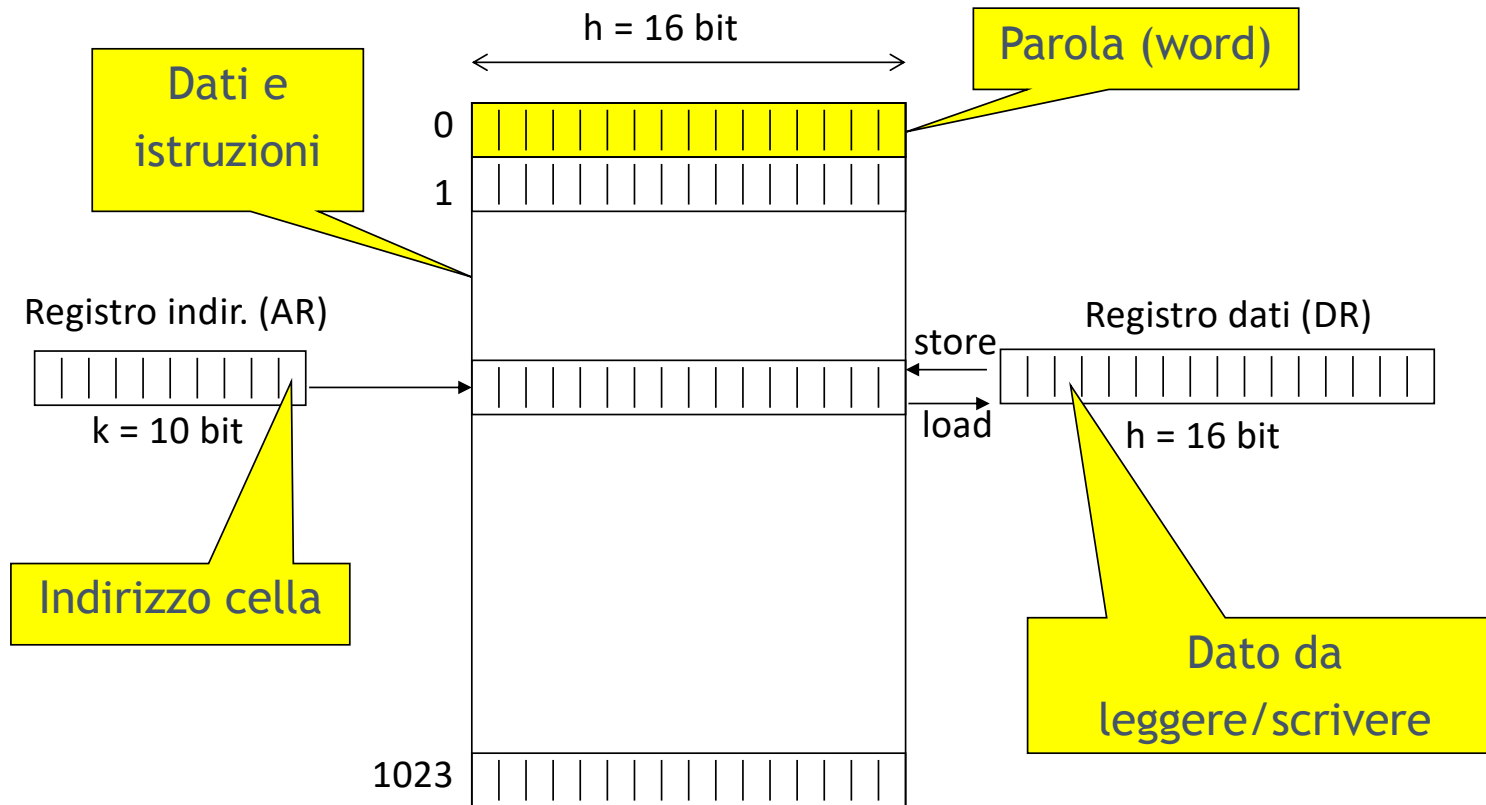


## Variabili

Una variabile è caratterizzata da:

- **Nome** → per distinguerla all'interno del programma
- **Tipo** → identifica il tipo di dato che è contenuto all'interno della variabile (numero intero, carattere, ...)
- **Indirizzo** → luogo fisico della memoria in cui la variabile è contenuta
- **Valore** → dato attuale

# Indirizzo di una variabile: la memoria del calcolatore



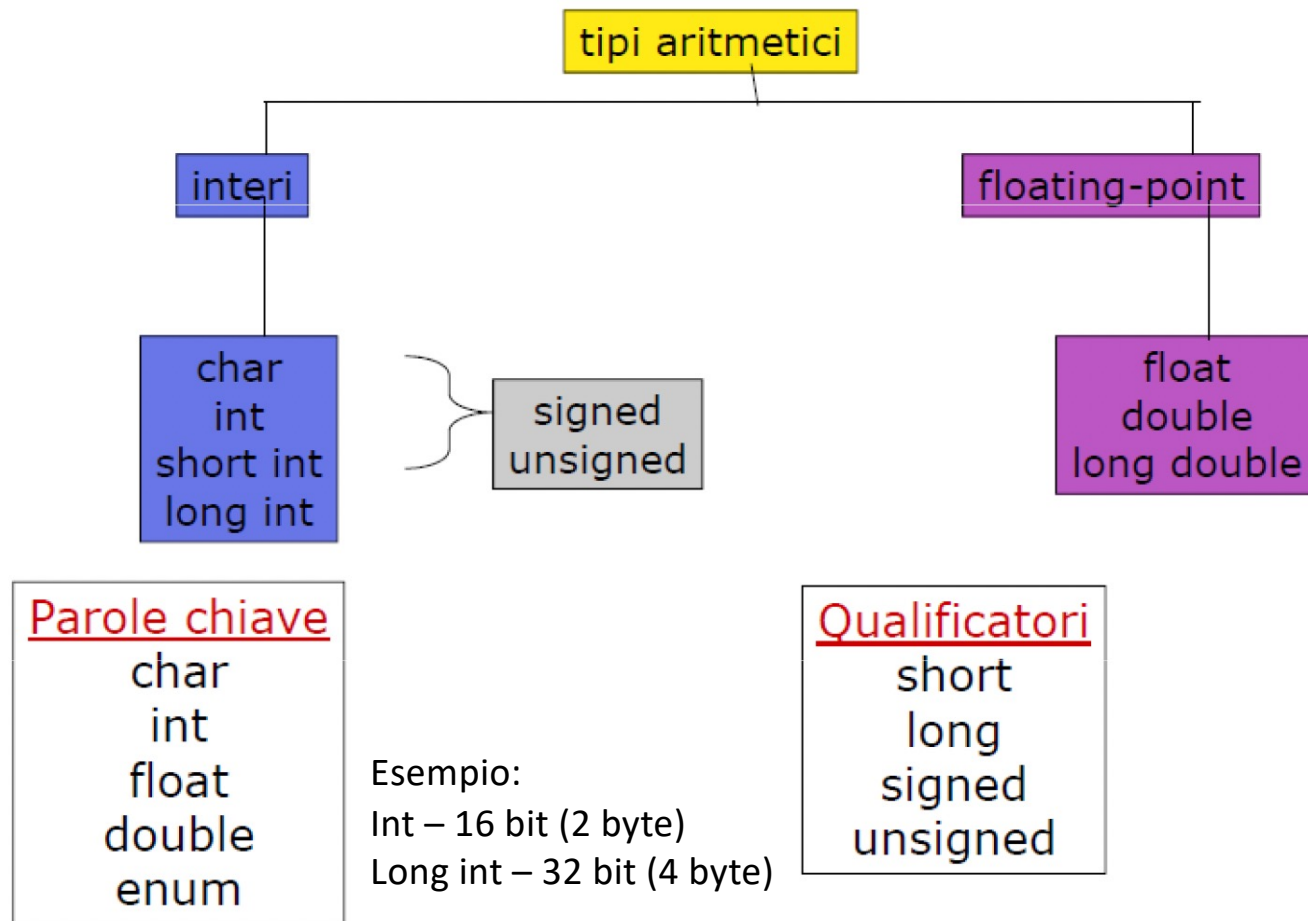
## Esempio: Una sequenza di istruzioni

Ipotizziamo che:

- si debba eseguire l'istruzione  
 $A \leftarrow A + B + C$   
(assegna alla variabile A la somma del contenuto delle variabili A, B e C);
- le corrispondenti istruzioni in linguaggio macchina si trovino all'indirizzo 789, 790, 791, ... (come riportato nella tabella sottostante);
- le variabili A, B e C si trovino rispettivamente nelle celle di memoria 4000 (A), 4004 (B) e 4008 (C).

Num	Istruzione	Commento
...	...	...
789	load R02,4000	trasferisce il contenuto della cella 4000 (A) nel registro R02
790	load R03,4004	trasferisce il contenuto della cella 4004 (B) nel registro R03
791	add R01,R02,R03	somma il contenuto dei registri R02 e R03 e scrive il risultato in R01
792	load R02,4008	trasferisce il contenuto della cella 4008 (C) nel registro R02
793	add R01,R01,R02	somma il contenuto dei registri R01 e R02 e scrive il risultato in R01
794	store R01,4000	trasferisce il contenuto del registro R01 nella cella 4000 (A)
...	...	...

# Tipi di dato semplice

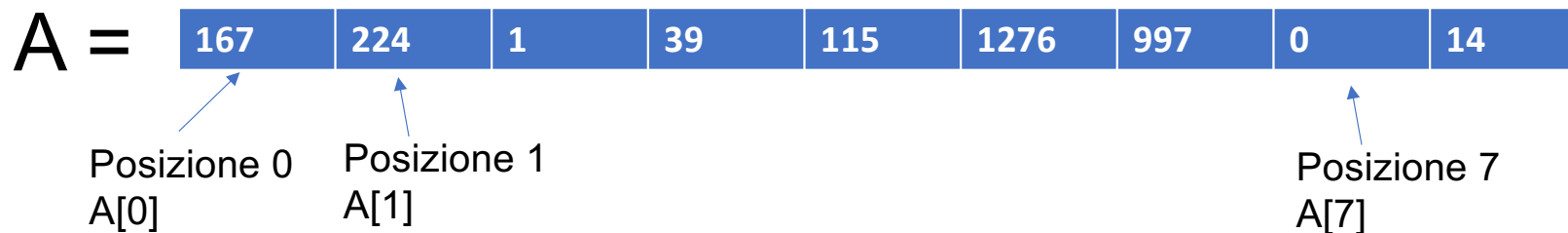


## Costanti

- La costante è una variable il cui contenuto non si modifica durante l'esecuzione
- Vengono utilizzate per migliorare la leggibilità del codice
- Per modificare un valore costante usato nel programma basta semplicemente cambiare la definizione della costante

## Tipi di dato strutturati: gli array

- Array (vettore) = sequenza numerata di oggetti
- In alcuni linguaggi gli oggetti devono essere dello stesso tipo (ad es C, Java), in altri no (ad es Python)
- Gli oggetti contenuti nell'array vengono raggiunti in base alla posizione in cui si trovano (indice)



## Tipi di dato strutturati: gli array multidimensionali

- Se in un elemento dell'array è contenuto un array a sua volta avrò un "array di array" o "matrice"
- Il numero di dimensioni rappresentabili è potenzialmente infinito

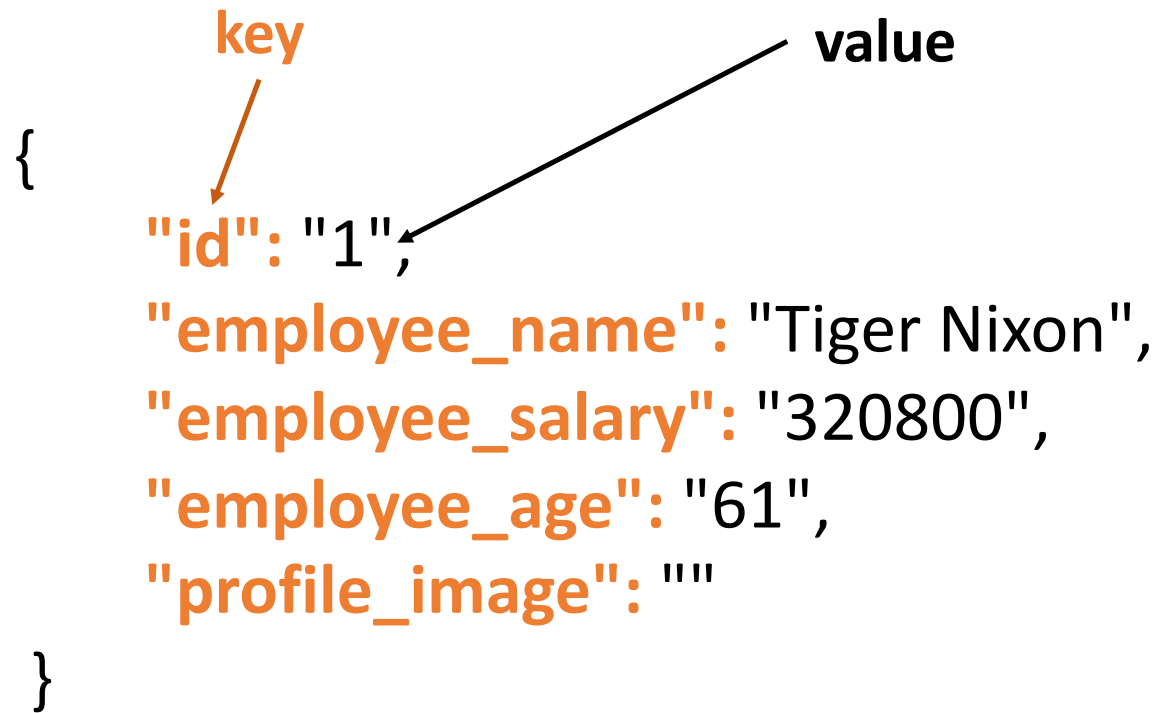
Col 0 = Array      Col 2      ...

167	224	1	39	115	1276	997	0	14	Riga 0 = Array
337	12	155	14	0	654	902	715	11	Riga 1
190	2015	777	934	652	678	131	16	1209	Riga 2

$A[0][2] = 1$

# JSON – JavaScript Object Notation

Formato utilizzato per lo scambio di dati in rete basato sulla coppia  
**chiave = valore**  
**key = value**



The diagram shows a JSON object with several key-value pairs. An orange arrow labeled 'key' points to the string 'id' in the first pair. A black arrow labeled 'value' points to the string '1' in the same pair. The rest of the object is shown in black text.

```
{  
  "id": "1",  
  "employee_name": "Tiger Nixon",  
  "employee_salary": "320800",  
  "employee_age": "61",  
  "profile_image": ""  
}
```



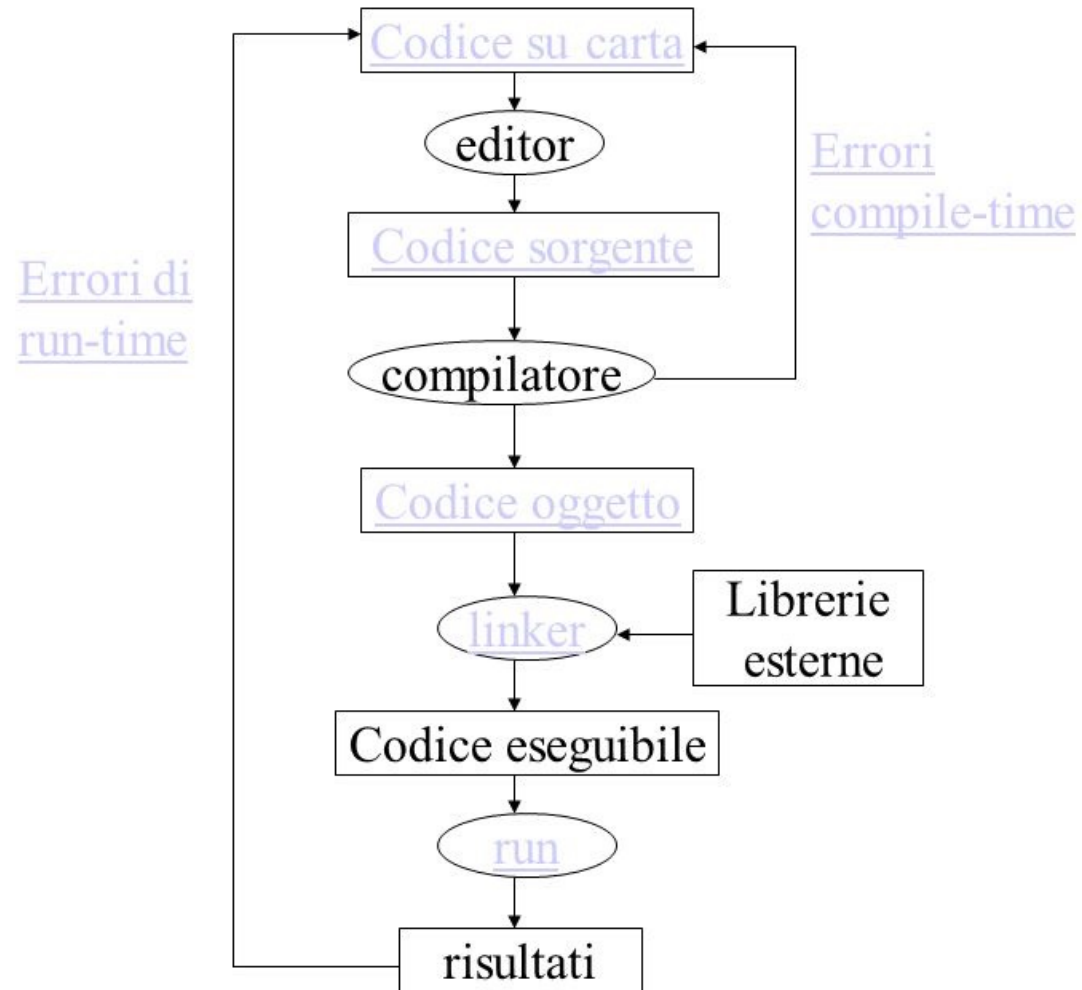
# ESEMPI

```
{
  "id": "1",
  "employee_name": "Tiger Nixon",
  "employee_salary": "320800",
  "employee_age": "61",
  "profile_image": ""
}, {
  "id": "2",
  "employee_name": "Garrett Winters",
  "employee_salary": "170750",
  "employee_age": "63",
  "profile_image": ""
}, {
  "id": "3",
  "employee_name": "Ashton Cox",
  "employee_salary": "86000",
  "employee_age": "66",
  "profile_image": ""
},
```

```
{
  "id": "4",
  "employee_name": "Cedric Kelly",
  "employee_salary": "433060",
  "employee_age": "22",
  "profile_image": ""
}, {
  "id": "5",
  "employee_name": "Airi Satou",
  "employee_salary": "162700",
  "employee_age": "33",
  "profile_image": ""
}, ...
```

**Il valore non si raggiunge  
attraverso la posizione nel json  
ma attraverso la chiave**

# Le fasi della programmazione

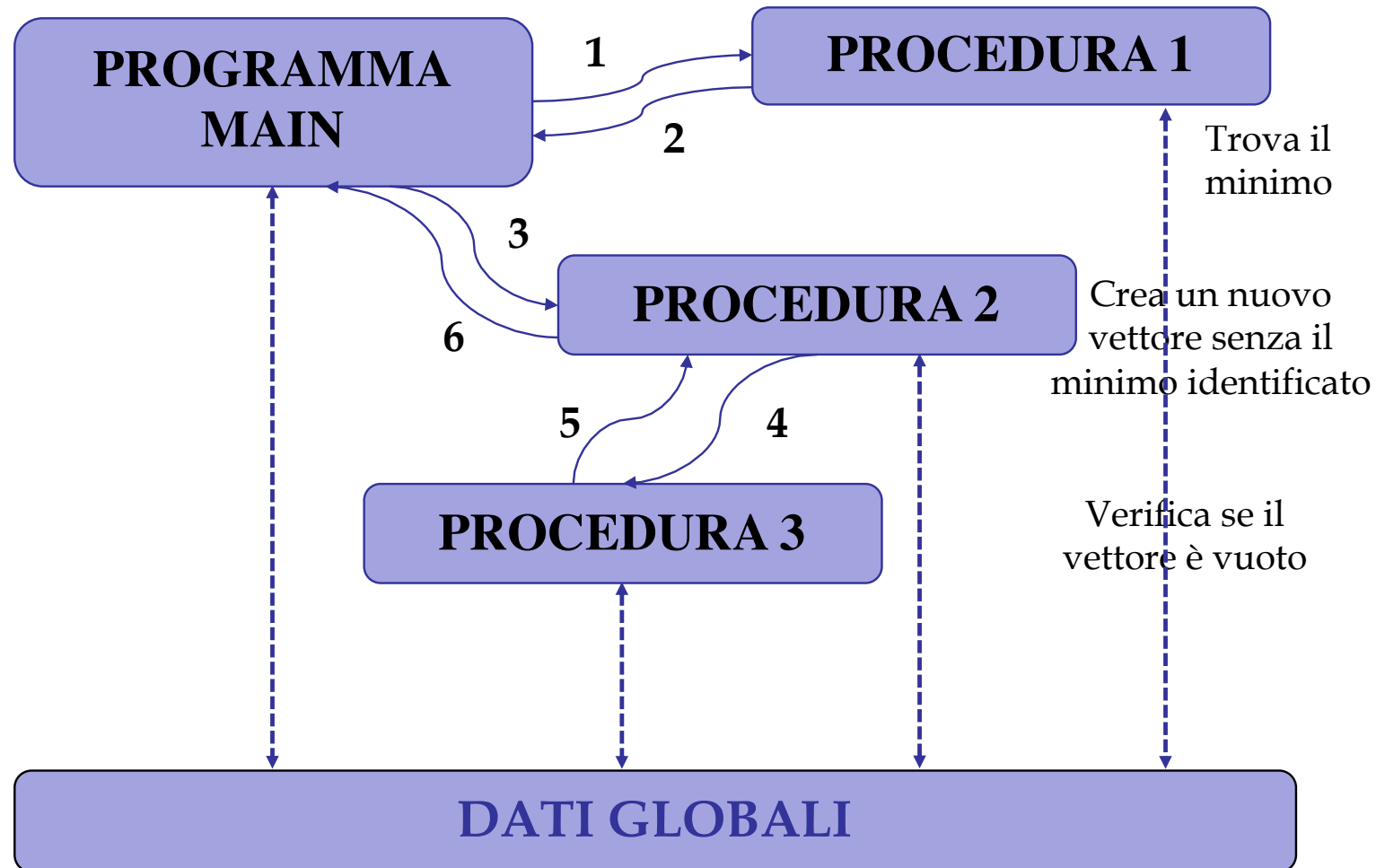


## Tipi di programmazione: Programmazione non strutturata

**STRUTTURE  
DATI**

**PROGRAMMA  
MAIN**

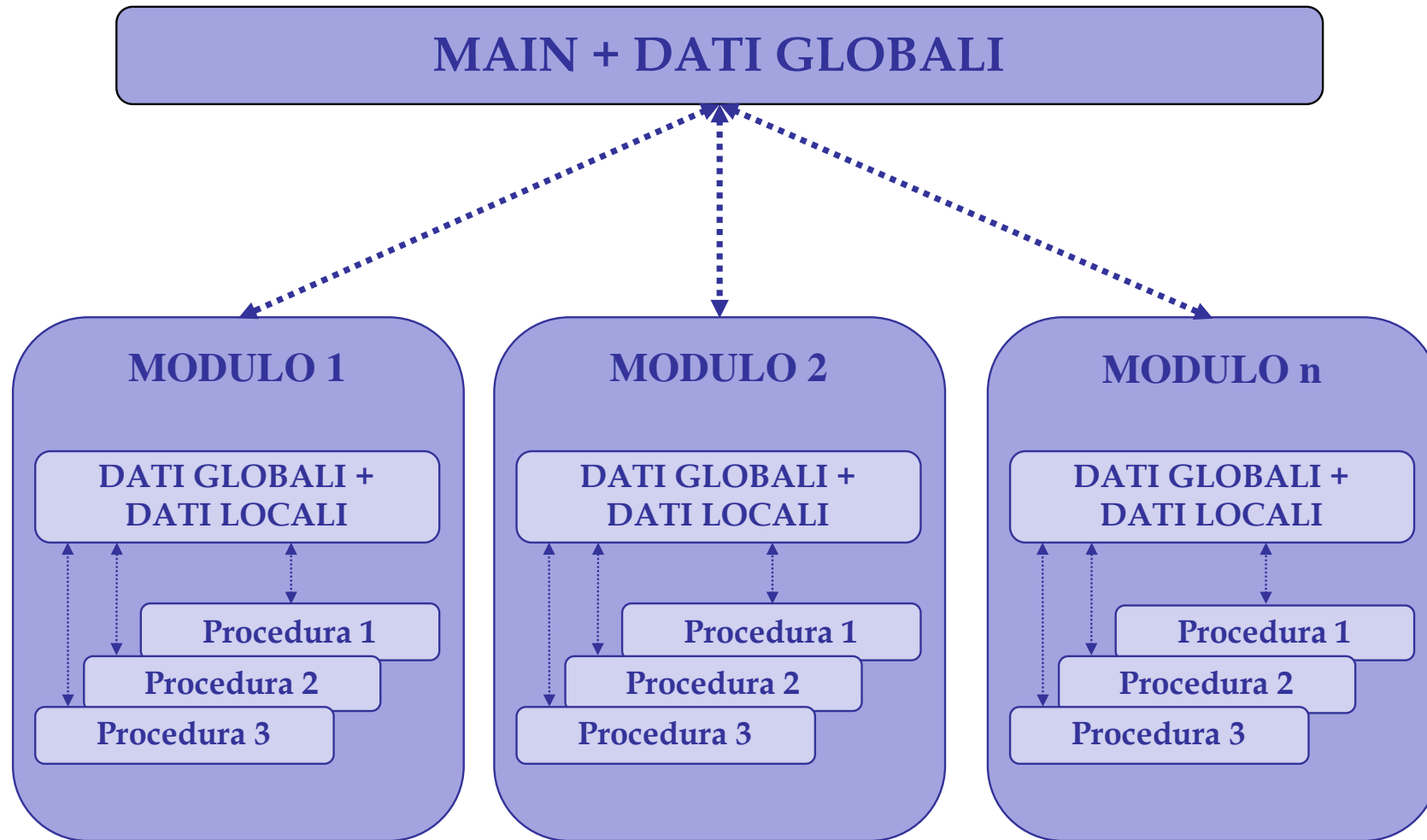
## Tipi di programmazione: Programmazione procedurale



## Programmazione procedurale

- Una procedura viene invocata col suo nome
- Una procedura accetta delle variabili che prendono il nome di **argomenti**
- Esempi:
  - LeggiDatiDaFile('c:\pippo.dat')
  - StampaSuVideo('salve mondo')
  - Addiziona(totale, nuovodato)

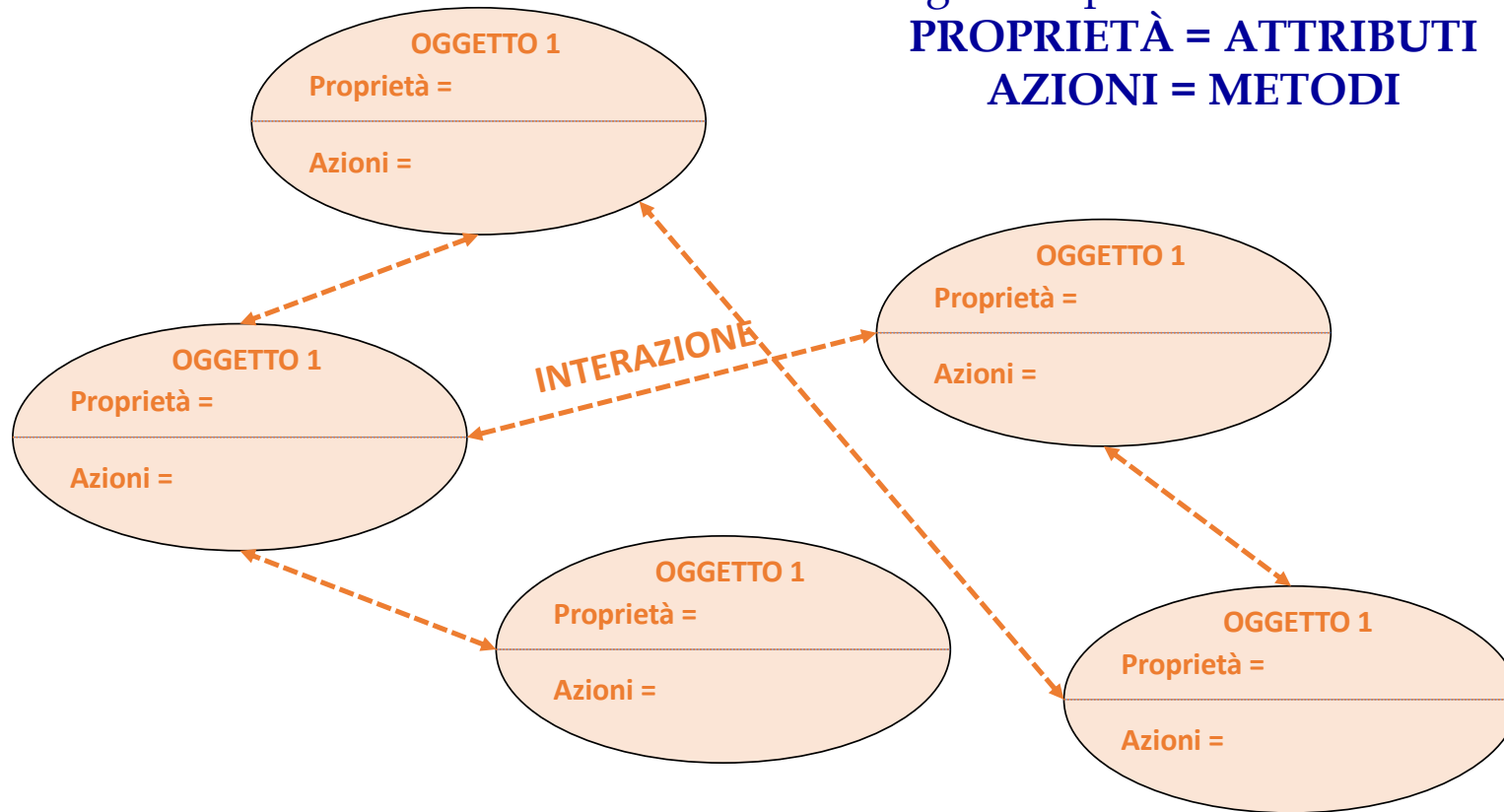
# Tipi di programmazione: Programmazione modulare



# Tipi di programmazione: PROGRAMMAZIONE A OGGETTI

## SISTEMA

Un sistema è un insieme di  
entità interagenti = OGGETTI  
in cui ogni componente è caratterizzato da  
**PROPRIETÀ = ATTRIBUTI**  
**AZIONI = METODI**



# ALGORITMI vs OGGETTI

## PROGRAMMAZIONE ALGORITMICA

- Sequenza di azioni
- Basato su DATI e FUNZIONI = PROGRAMMI
- Obiettivo: risolvere un PROBLEMA

## PROGRAMMAZIONE A OGGETTI

- Sistema = Insieme di oggetti
- Basato su OGGETTI fatti da AZIONI e ATTRIBUTI
- Obiettivo: gestire un SISTEMA



# OGGETTI E CLASSI



# OGGETTI E CLASSI



Classi

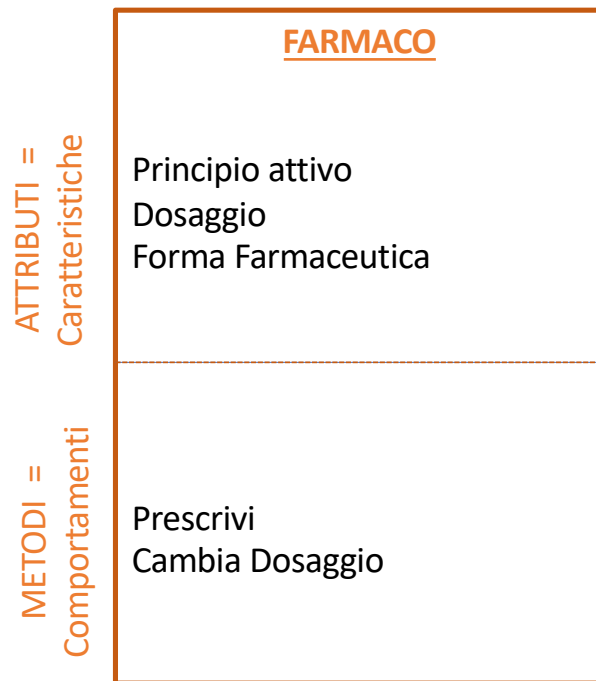
Istanze (Oggetti)

Orco

Cavaliere

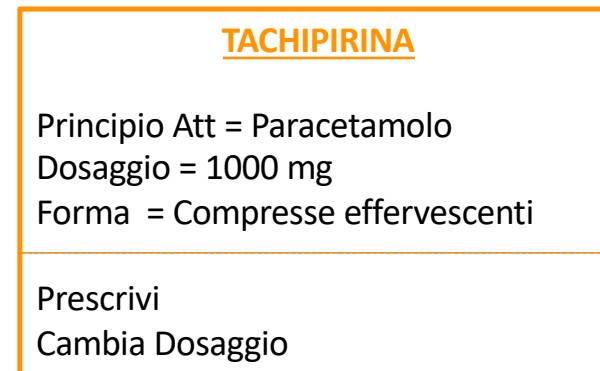
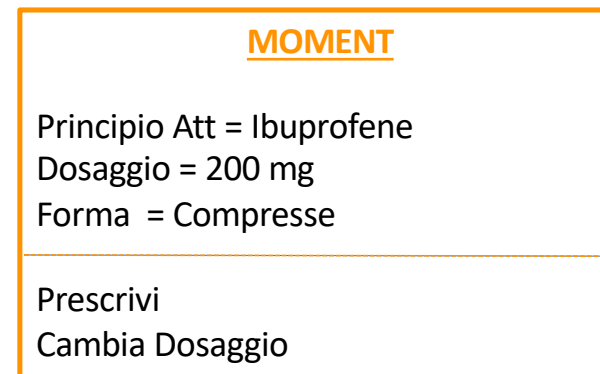
# ESEMPIO

## CLASSE



**OGGETTO =  
ISTANZA DI UNA CLASSE**

## OGGETTI



## ESEMPIO: OSSERVAZIONI

### MOMENT

Principio Att = Ibuprofene  
Dosaggio = 200 mg  
Forma = Compresse

Prescrivi  
Cambia Dosaggio

I VALORI DEGLI ATTRIBUTI  
SONO SPECIFICI  
DELL'OGGETTO ISTANZIATO  
(ogni oggetto ha il suo insieme di  
valori)

LE AZIONI SONO COMUNI A  
TUTTE LE ISTANZE DELLA  
CLASSE

# ATTRIBUTI E METODI

## ATTRIBUTI

- Descrivono le proprietà **statiche** dell'oggetto
- Nella programmazione gli attributi vengono realizzati attraverso l'uso delle **variabili** utilizzate dall'oggetto per memorizzare i dati

## METODI

- Descrivono le proprietà **dinamiche** dell'oggetto
- Nella programmazione i metodi vengono realizzati attraverso la scrittura di codice (**procedure e funzioni**) che implementano le operazioni dell'oggetto



# INTERAZIONE TRA OGGETTI: MESSAGGI

- Un programma ad oggetti è caratterizzato dalla presenza di tanti oggetti che **interagiscono fra loro attraverso il meccanismo dello scambio di messaggi**
- I messaggi possono:
  - Richiedere un'informazione su un oggetto
  - Modificare lo stato di un oggetto

