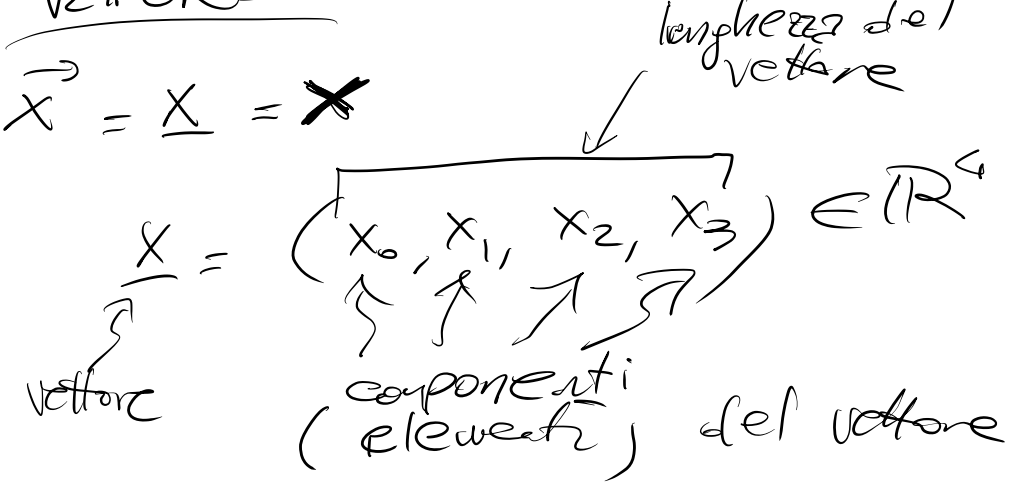
ARRAY & SLICE

- "VETORE"

$\vec{x} = \underline{x} = \mathbf{x}$

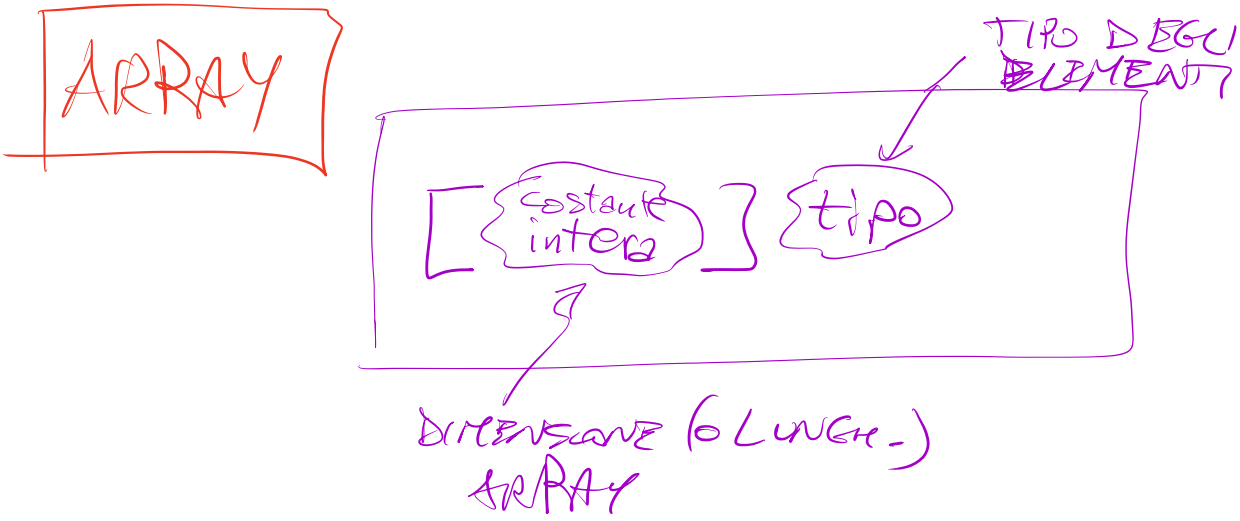


```
var x, s, n int
fmt.Scan(&n)
for i := 0; i < n; i++ {
        fmt.Scan(&x)
        s += x
}
media := float64(s) / float64(n)
fmt.Println(media)
```

$$\sigma^2 = Var \overset{\mu}{\overset{\Delta}{=}} \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n} =$$

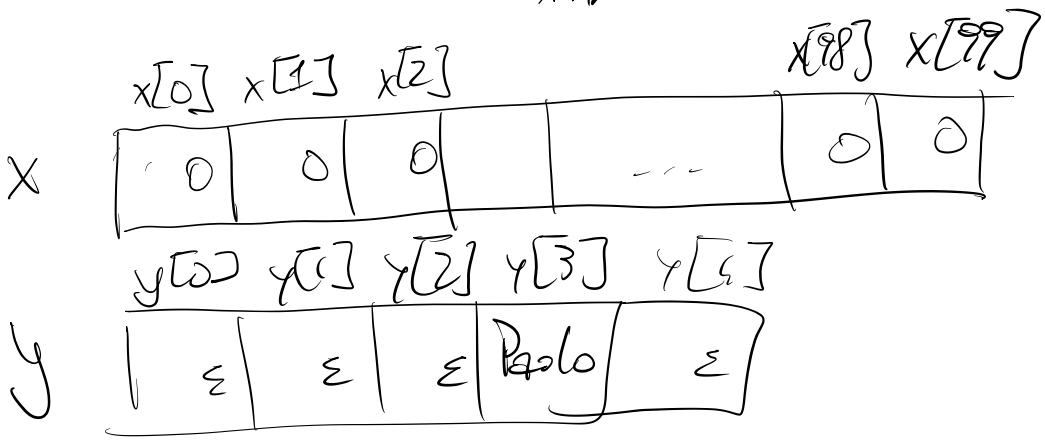
$$= \frac{(x_1-\mu)^2 + (x_2-\mu)^2 + \dots + (x_n-\mu)^2}{n}$$

ARRAY

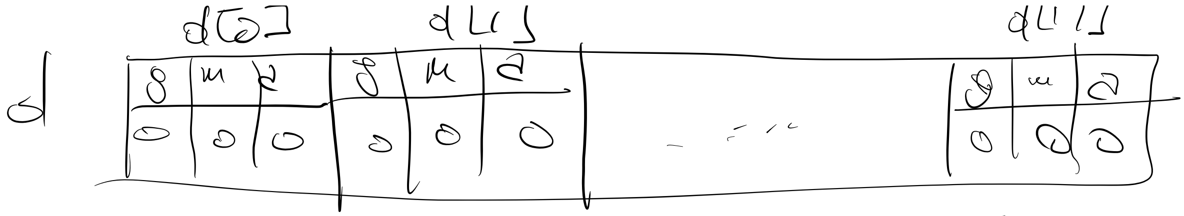



```
var   x   [100] int
var   y   [5]   string
var   d   [20]  data
var   p   [10]  * int
```

COSTANTI



y[3] = "Paolo"



d[0] = data{29, 11, 1968}

oppure

d[0].g = 29

d[0].m = 11

d[0].a = 1968

d[i]

≥0

< len(d)

```
var a [100] int
fmt.Scan(&n)
for i := 0; i < n; i++ {
    fmt.Scan(&a[i])
}
var s int
for i := 0; i < n; i++ {
    s += a[i]
}
media := float64(s) / float64(n)
var sq float64
for i := 0; i < n; i++ {
    sq += (float64(a[i]) - media) *
          (float64(a[i]) - media)
}
```

if n > len(a) {
    n = len(a)
}

Ciclo for

```
}
variazz := sq / f64(n)
sqm = math.Sqrt(variazz)
```

ARRAY E CICLI

I°
```
for i:=0; i<len(a); i++ {
    ... i ... a[i] ...
}
```

II°
```
for i := range a {
    ... i ... a[i] ...
}
```

III°
```
for i, x := range a {   // x serve solo per la lettura
    ... i ... x ... a[i] ...
}
```

"DIFETTI" DEGLI ARRAY

- DIMENSIONE STATICA
  ⇒ sono di dimensione fissa
- ASSEGNAMENTO = COPIA

```
var a [100]int
var b [100]int

b = a
```

- FUNZIONI CON ARGOMENTO UN ARRAY SENZA DIMENSIONE FISSA

```
func media(a [100]int, n int)
                    float64 {
...
}
```

}

LETTERALI ARRAY

```
x := [8] int {1, 2, 3, 4, 5}
```

x

1	2	3	4	5	0	0	0

```
x := [...] int {1, 2, 3, 4, 5}
```

x

1	2	3	4	5

```
x := [10] int {0: 3, 7: 5}
```

x

0	1	2	3	4	5	6	7	8	9
3	0	0	0	0	0	0	5	0	0

```
func printData(d data) {
    nome Mese := [...]string{"Gen",
        "Feb", "Mar", ...}

    printf("%d %s %d",
        d.g, nome Mese[d.m-1],
        d.a)

}
```

SLICE

- DIMENSIONE NON E' STATICA E PUO' VARIARE NEL TEMPO

[] tipo

ACH.

var	x	[] int
var	s	[] string
var	t	[] data
var	p	[] * int

} nil

len(x) ⇝ lunghezza attuale della slice

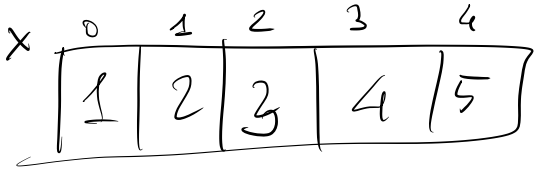
len (nil) = 0

x[0], x[1], ..., x[len(x) - 1]

CREAZIONE

- LETTERALI SLICE

```
x = []int{1, 2, 3, 4, 5}
```



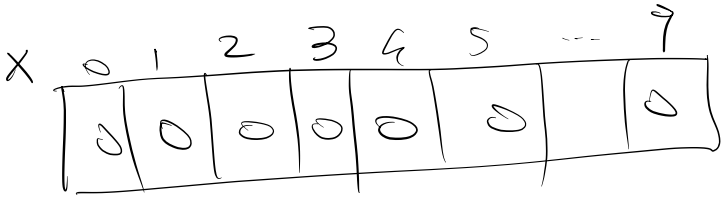
~~`x = [5]int{1, 2, 3, 4, 5}`~~

- FUNZIONE make

```
x = make([]int, 10)
```

TIPO DI SLICE DA CREARE

LUNGH.



```
var n int
fmt.Scan(&n)
var a []int
a = make([]int, n)
```
] → a := make([]int, n)

```
for i := 0; i < n; i++ {
    fmt.Scan(&a[i])
}
for _, x := range a {
    s += x
}
media := float64(s) / float64(n)
for _, x := range a {
    sq += (float64(x) - media) *
          (float64(x) - media)
}
varianza := sq / float(len(a))
```

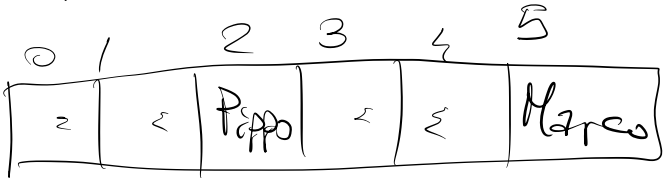
ALLUNGAMENTO SLICE

```
X = append(x, lo)
```

- `x` → SLICE DA ALLUNG.
- `lo` → NUOVO ELEMENTO

```
var s []string
s = []string{"Paolo", "Boldi"}
s = append(s, "Anna")
s = append(s, "Marco")
```

```
var s []string
s = make([]string, 7)
s = append(s, "Paolo")
```




```
s[2] = "Pippo"
s[5] = "Marco"
```

- ESERCIZIO: leggere una sequenza di nomi ristamparli al contrario e stampare il nome più lungo