MEMORIA IN GO

1) MEMORIA STATICA
2) STACK (DI ESECUZIONE)
3) HEAP

① Oggetti persistenti (variabili globali)

② Tutti gli oggetti allocati con new/make (garbage collection)

③ Di norma si trovano le variabili locali (automatiche)

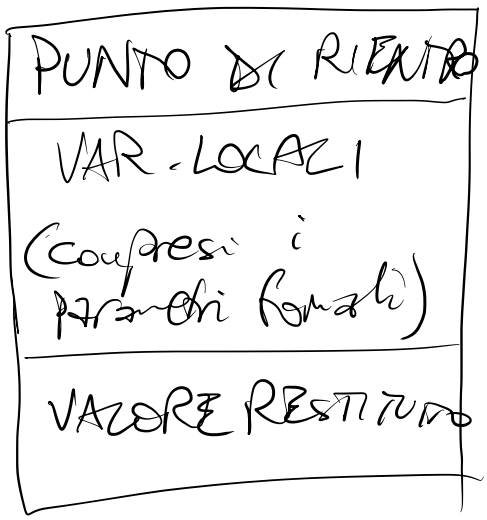
STACK (Pila)

- Struttura dati

LIFO = Last In First Out

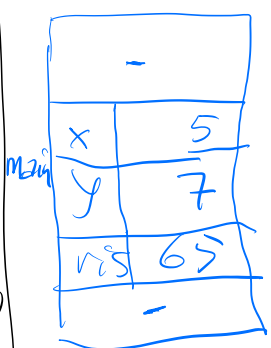
STACK DI ESECUZIONE

RECORD DI ATTIVAZ. (RA)



```
1   func main() {
2       var x,y,ris int
3       fmt.Scan(&x)
4       fmt.Scan(&y)
5   →   ris = f(x,y)
6       fmt.Printf("%d\n", ris)
7   }

8   func f(a,b int) (c int) {
9       var x,y int
10      x = sqr(a)
11      y = sqr(b)
12  →   c = x+y+1
13      return
14  }
15  func sqr(x int) (z int) {
16      z = x*x
17      return
18  }
```



RICORSIONE

- Definizione ricorsiva: è una definizione in cui si usa l'oggetto che si sta definendo

DEF 1 (iterativa)

$$n! \triangleq 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

DEF 2 (ricorsiva)

$$n! \triangleq \begin{cases} 1 & \text{se } n = 0 \\ n \cdot (n-1)! & \text{altrimenti} \end{cases}$$

$3! = 3 \cdot (2)! =$

$= 3 \cdot 2 \cdot (1)! =$

$= 3 \cdot 2 \cdot 1 \cdot (0)! =$

$= 3 \cdot 2 \cdot 1 \cdot 1 = 6$

```
func fattoriale (n int) int {
    prod := 1
    for i := 1; i <= n; i++ {
        prod *= i
    }
    return prod
}
```

```
func fattoriale(n int) int {
    if n==0 {
        return 1
    } else {
        return n * fattoriale(n-1)
    }
}
```

$$n! \triangleq \begin{cases} 1 & \text{se } n=0 \\ n \cdot (n-1)! & \text{altrimenti} \end{cases}$$

- FIBONACCI

$$F_0 = F_1 \triangleq 1$$

$$F_n = F_{n-1} + F_{n-2}$$

n	F_n
0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21
...	...

funz fib(n int)