GESTIONE DELLA MEMORIA

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

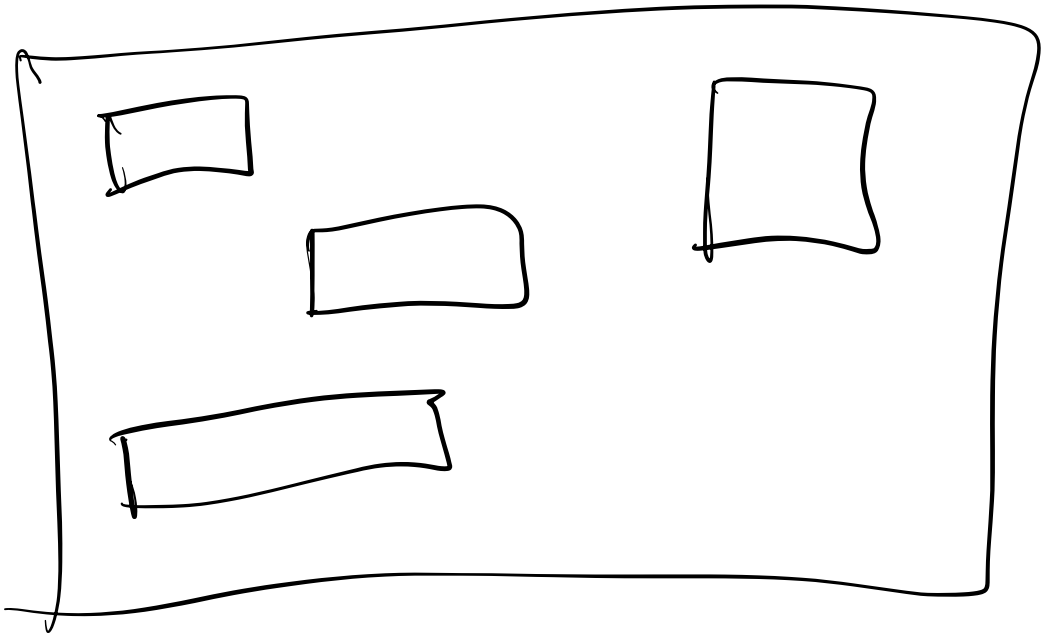
1. Vivono gli oggetti che devono esistere per tutta l'esecuz. del programma

pippo.go

```
var x,y string

func main() {
    var a,b int
    ⋮
}
```

3 HEAP

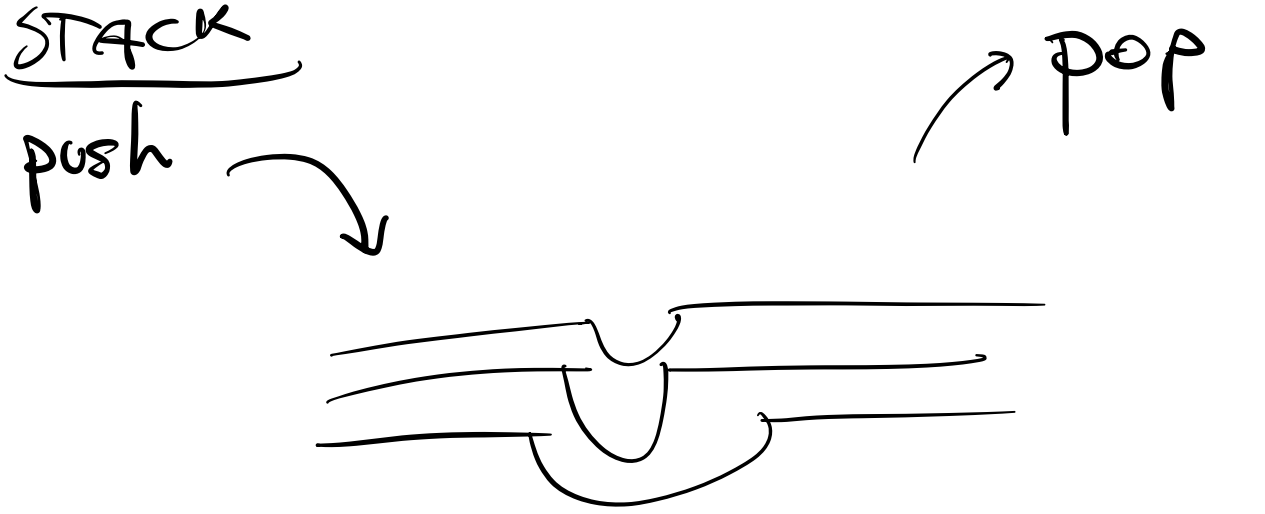
new / make



2 STACK DI ESECUZIONE

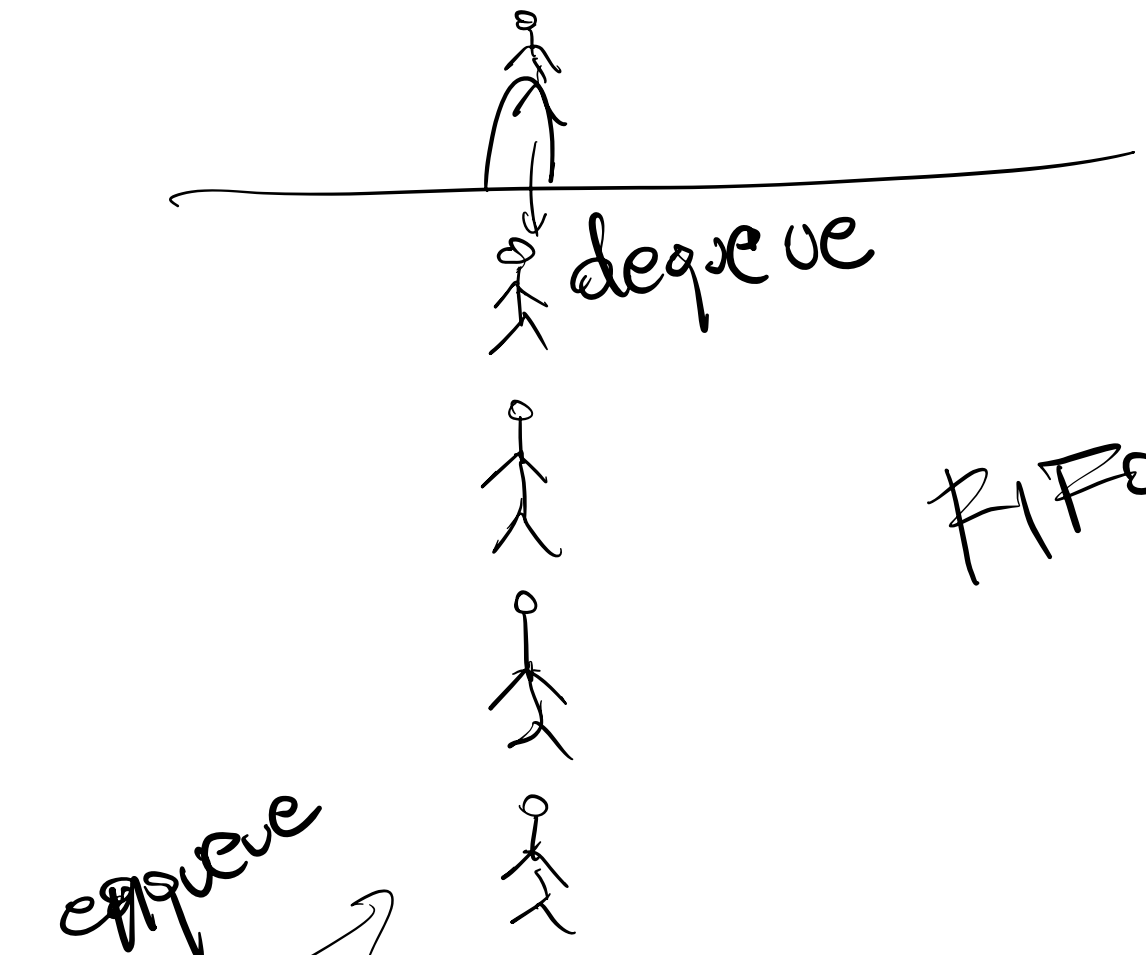
STRUTTURA DATI

STACK



LIFO

CODA

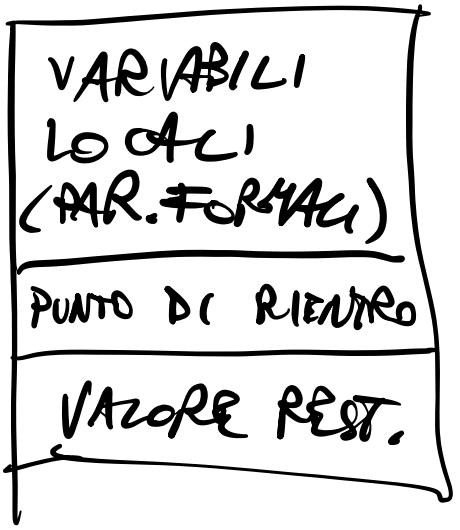


FIFO

RECORD DI ATTIVAZIONE

```
func f(x, y int){
    var a, b, ...

}
```

R.A.



```
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) (a int){
16  →   a = x*x
17      return
18  }
```

```
3 ↵
7 ↵
59 ↵
```

RICORSIONE

$$n! = 1 \cdot 2 \cdot 3 \cdot \underbrace{\ldots} \cdot n$$

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

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

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

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

$$n?! = \begin{cases} 1 & \text{se } n=0 \\ n+3\ (n+1)?! & \text{alt.} \end{cases}$$

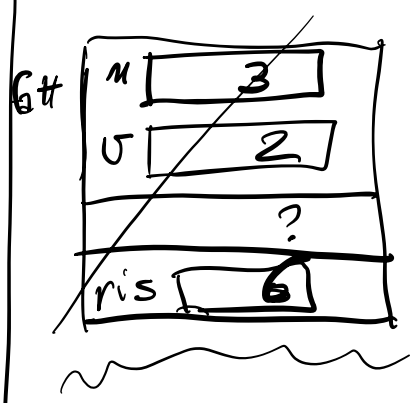
$$3?! = 3+3\cdot 4?! =$$

$$= 3+3\ (3+3\cdot 5?!) =$$

$$= \ldots$$

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

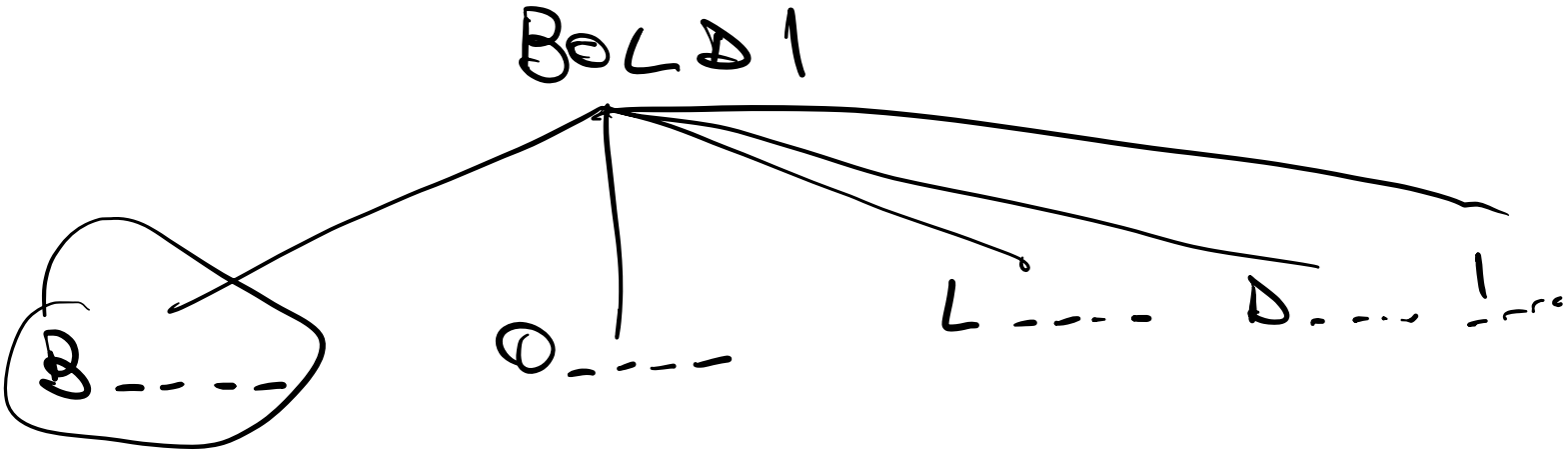
```
1 func  fatt (n int) (ris int){
2       if  n==0 {
3           ris=1
4       }else {
5           v:= fatt(n-1)
6           ris= n*v
7       }
8       return
9 }

fatt(3)
```



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

ANAGRAMMI

BOLDI




```
func anagrammi(s string) []string {
    if s == "" {
        return []string{""}
    } else {
        var ris []string
        for i := 0; i < len(s); i++ {
            primo := rune(s[i])
            resto := s[:i] + s[i+1:]
            anagResto := anagrammi(resto)
            for _, x := range anagResto {
```

```
        ris = append(ris,
          string(primo)+x)
    }
}
return ris
}
}
```

```
func f (x int) int {
    if x < 10 {
        return 1
    } else {
        return 1 + f(x/10)
    }
}
```

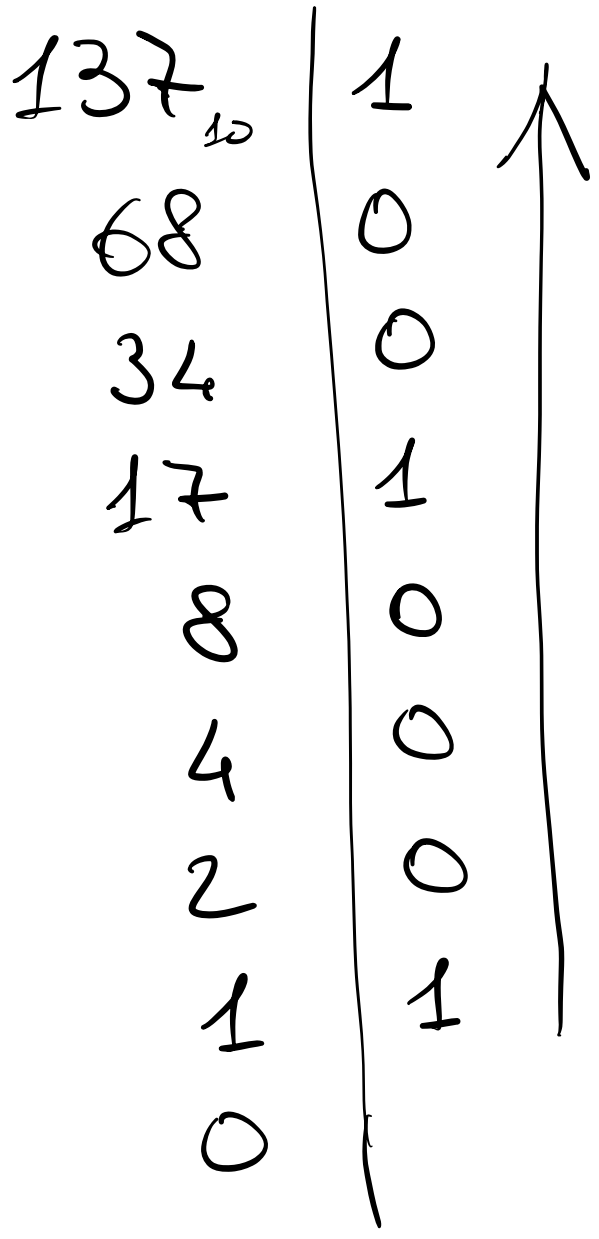
$f(1357) = 1 + f(135) =$

$= 1 + 1 + f(13) =$

$= 1 + 1 + 1 + f(1) =$

$= 1 + 1 + 1 + 1 =$

$= 4$



1000 1001

```
func printBinary(x int) {
    if x < 2 {
        fmt.Print(x)
    } else {
        resto := x % 2
        printBinary(x/2)
        fmt.Print(resto)
    }
}
```