DEFINIZIONI DI TIPO

DEFINIZIONE DI TIPO

type {id} {tipo}

↑

TIPO SOGGIACENTE

---

type punt **float64

var x, y, z punt

var p *punt

```
type   carta   int
```

```
func   seme (c carta)   int
```

```
type   carta   int
var (
    i   int
    c   carta
)
i=c //NO
c=i //NO
```

{ i = int(c)
  c = carta(i)

package main

import (

...

)

type (
carta int
seme int
)

func qualeSeme(c carta) seme {

...

}

↑
Più PROTETTIVO

ALIAS DI TIPO

SI USANO MOLTO, POCO

type id = tipo

---

type intero = int64

type
var (
x int64
y intero
)

x = y // SI
y = x // SI

STRUCT

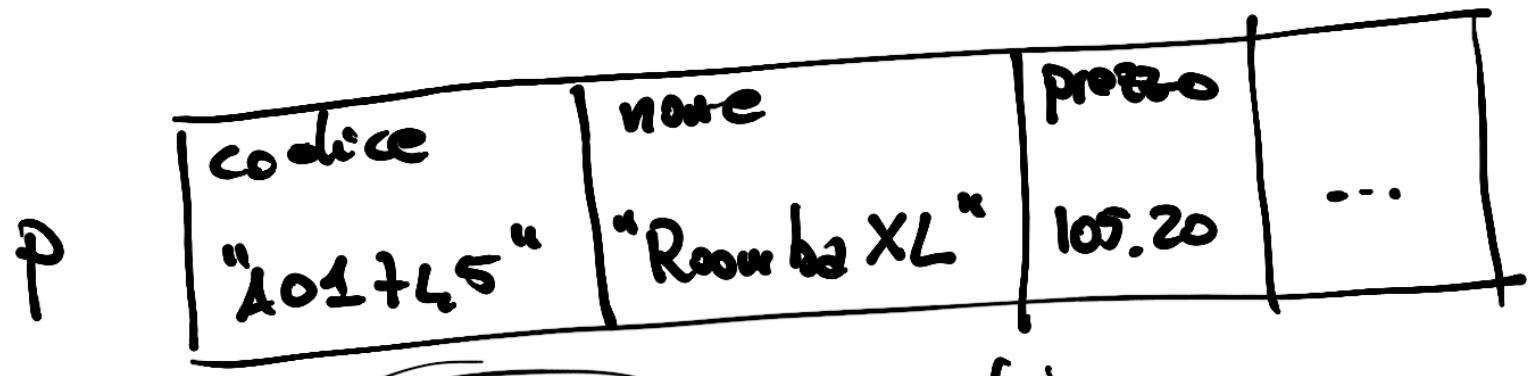
- TIPO COSTITUITO DA TANTI ELEMENTI

DATA

```
Var d struct {
        g int
        m int
        a int
    }
```

```
type data struct {
        g  int
        m  int
        a  int
}
var d1, d2, d3  data
func  f (d data) int {
        ...
}
var  p  *data
```

```
type prodotto struct {
    codice string
    nome string
    prezzo float64
    inMagazzino bool
    ...
}

var p prodotto
```

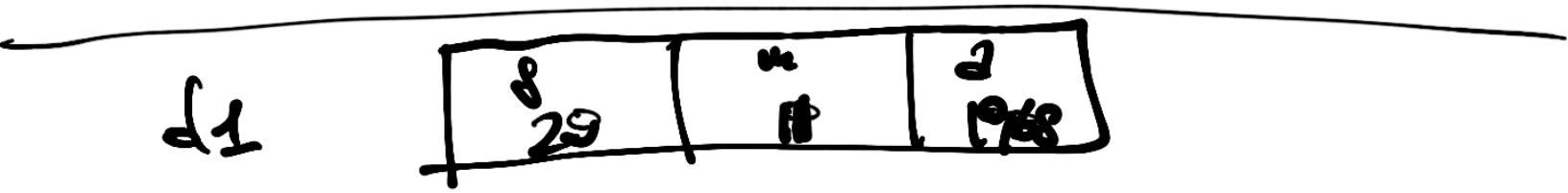


p.codice string

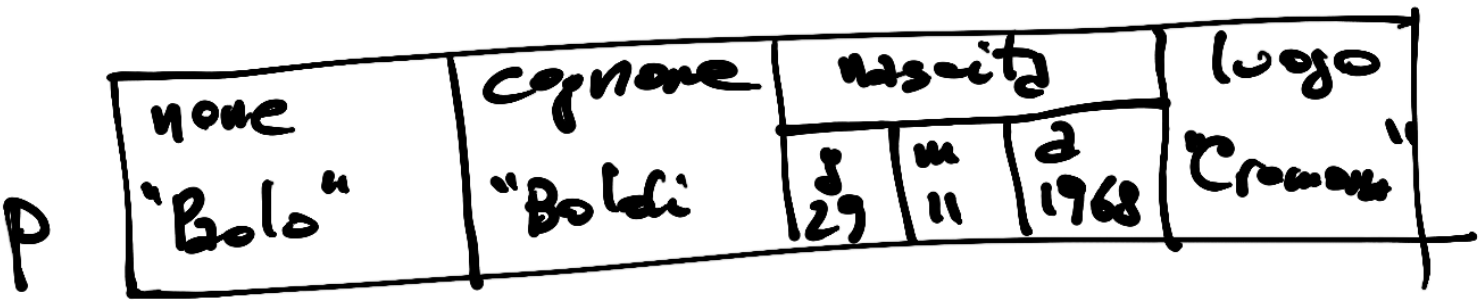
p.nome string

p.prezzo float64

```
type data struct {
    g, m, a int
}

var d1, d2 data
d1.g = 29
d1.m = 11
d1.a = 1968
d2.g, d2.m, d2.a = 18, 10, 2019
d2 = data{18, 10, 2019}
d2 = data{m: 10, a: 2019}
     ↑
```

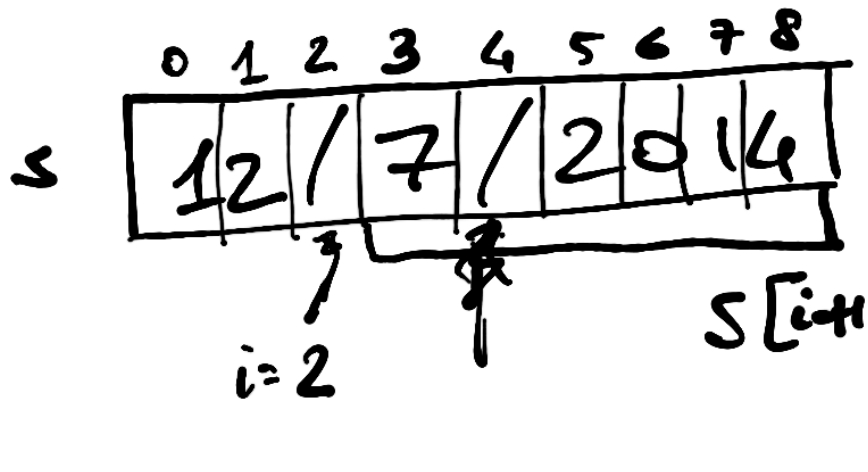


```
type (
    data    struct {
                g, m, a  int
            }
    persona struct {
                nome, cognome  string
                nascita        data
                luogo          string
            }
)
var p persona
```




```
p.nome
p.nascita.g
p.nascita.a
```

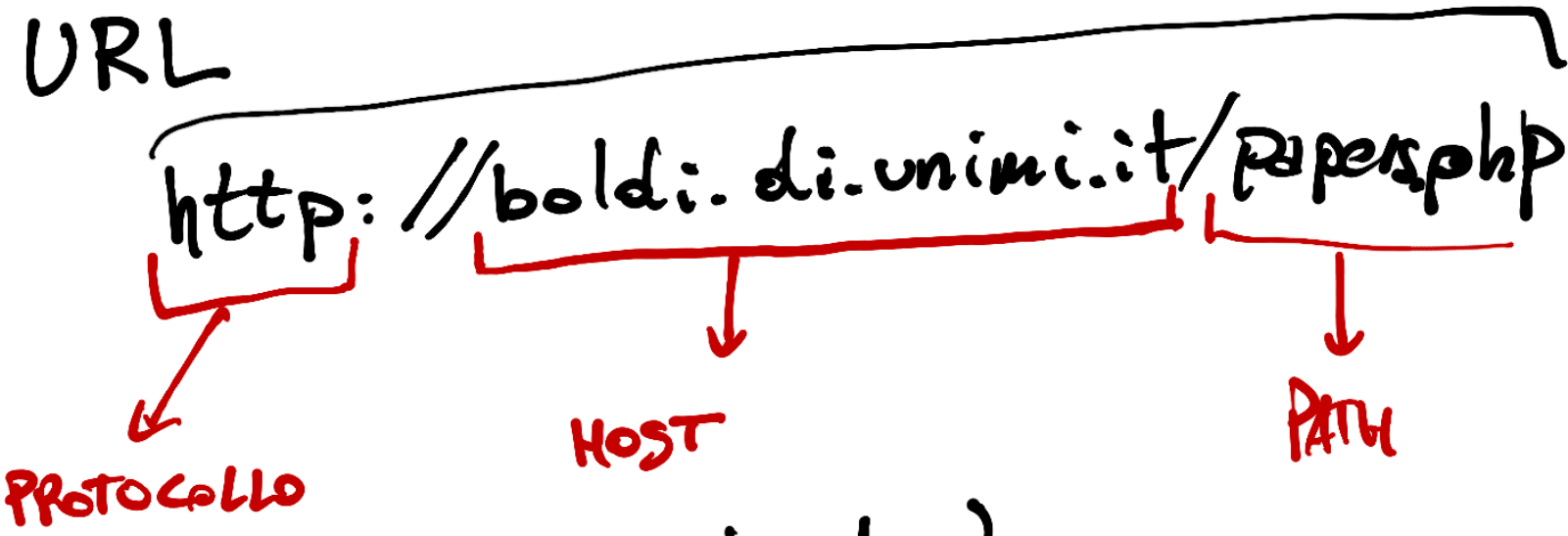
```
func isLeap(a int) bool {
    return a%4==0 && (a%100!=0
        || a%400==0)
}

func daysFromStart(d date) int {
    var c int
    for i:=1; i<d.m; i++ {
        switch(i) {
            case 11, 4, 6, 9:
                c+=30
            case 2:
                if isLeap(d.a) {
                    c+=29
                }else{
                    c+=28
                }
            default:
                c+=31
        }
    }
    return c+d.g
}
```




```
func string2data(s string) data {
→ i := strings.Index(s, "/")
  j := strings.Index(s[i+1:], "/")
  gs := s[:i]
  ms := s[i+1 : i+1+j]
  as := s[i+1+j+1:]
  g, _ := strconv.Atoi(gs)
  m, _ := strconv.Atoi(ms)
  a, _ := strconv.Atoi(as)
  return data{g, m, a}
}
```

```
var d data
fmt.Scan(&(d.g))
fmt.Scan(&(d.n))
fmt.Scan(&(d.d))
```

URL




```
type url struct {
    prot string
    host string
    path string
}
```

① func string2url(s string) url(

② STESSA FUNZIONE MA LA FUNZIONE SCRIVE DIRETT. IN UNA VARIABILE DI TIPO url(

- SCRIVETE UNA FUNZIONE CHE DATO UN ANNO RESTITUISCE IL GIORNO DI NATALE DI QUELL'ANNO

```
func natale(a int) data {
    return data{25,12,a}
}
```

```
func natale(a int) data {
    var v data
    v.g = 25
    v.m = 12
    v.a = a
    return v
}
```

```
func natale2(a int, d *data) {
    (*d).g = 25
    (*d).m = 12
    (*d).a = 2
}
```

```
var d data
d = natale(2019)
```

```
var d data
natale2(2019, &d)
```

```
var d1, d2 data
d1 = d2
```

```
func tostring(d data) string {
    var s string
    if d.g < 10 {
        s = "0"
    }
    s += strconv.Itoa(d.g)
    s += "/"
    if d.m < 10 {
        s += "0"
    }
    s += strconv.Itoa(d.m)
    s += "/"
    t := strconv.Itoa(d.a)
    for len(t) < 4 {
        t = "0" + t
    }
    return s+t
}
```

PADDING