```
func inc(x int) {
    x++
}

func main() {
    var a int
    a = 5
    inc(a)
    fmt.Println(a) <- 5
}
```

```
func inc(x *int) {
    (*x)++
}

func main() {
    var a int
    a = 5
    inc(&a)
    fmt.Println(a) <- 5
}
```

x: 0x79735

a: 0x79735 → 6

# TYPE

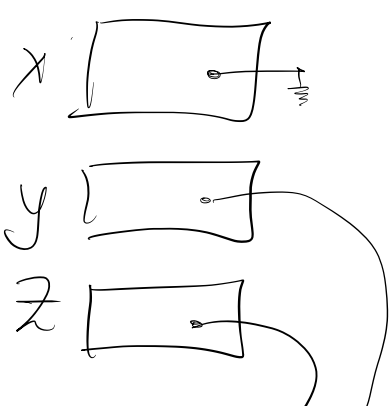
**type** {id} {tipo}

QUASI IDENTICO A **var**

| | | |
|---|---|---|
| **var** | x | *int |
| **type** | p | *int |
| **var** | y, z | p |
| **var** | a | int |

definizione di tipo

```
a = 5
y = &a
z = y
*y = *z + 3
```



fmt.Println(a)

a $\boxed{8}$

$y = x$

No

$\rightsquigarrow$ $y = P(x)$

STRUCT ≈ record

type data struct {

g, m, a int

}

campi

var x, y data

x

| g | m | a |
|---|---|---|
| 0 | 0 | 0 |

y

| g | m | a |
|---|---|---|
| 0 | 0 | 0 |

struct {

dichiarazione di campi

}

var x struct {

g, m, a int

}

var y struct {

g, m, a int

}

```
package main
import (
    ...
)
type data struct {
    g, m, a
}

func f()  ...
func g()  ...
func main()  ...
```

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 = 9, 11, 2021

type studente struct {
nome, cognome string
dataNascita data
matricola string
}

var s1, s2 studente

s1

| nome | cognome | dataNascita | | | matricola |
|---|---|---|---|---|---|
| | | g | m | a | |
| | | | | | |

s1.nome = "Paolo"

s1.cognome = "Boldi"

s1.dataNascita.g = 29

s1.dataNascita.m = 11

s1.dataNascita.a = 1968

s1.matricola = "352372"

d2 = d1

---

**LETTERALI struct**

d1 = data {29, 11, 1968}

s1 = studente {"Mario", "Rossi",
data {3, 11, 1995},

"374956" }

s2 = student { name: "Maria",
dataNascita: data{3,7,1963}
}

- Funzione che restituisce il Natale di un dato anno

```
func christmas(anno int) data {
    var d data
    d = data{25, 12, anno}
    return d
}

func christmas(anno int) data {
    return data{25, 12, anno}
}
```

- Scrivere una funzione che modifica una data lasciando invariati mese e anno e ponendo il giorno uguale a 1.

```
func giornoUno(d *data) {
    (*d).g = 1
}

func main() {
    var x data
    x = data{9, 11, 2021}
    giornoUno(&x)
    fmt.Println(x)
}
```

- Funzione che stampa una data

```
func printData(d data) {
    fmt.Printf("%02d/%02d/%d",
        d.g, d.m, d.a)
}
```

␣9/11/2021

09/11/2021

%d

%2d

%02d

$\boxed{\text{9 Nov 2021}}$

```
func printData2(d data) {

    var m string
    switch d.m {
        case 1:  m = "Gen"
        case 2:  m = "Feb"
        ...
        case 12: m = "Dic"
    }
    fmt.Printf("%d␣%s␣%d",
        d.g, m, d.a)
}
```

- Funzione che data una stringa del tipo
xxx/xxx/xxxx
restituisca la data corrispondente

```
func convData(s string) (data, bool) {
    p1 := strings.IndexRune(s, '/')
    if p1 == -1 {
        return data{}, false
    }
    p2 := strings.IndexRune(
        s[p1+1:], '/')
    if p2 == -1 {
        return data{}, false
    }
    p2 += p1+1
    var d data
    var err error
```

```
d.g, err = strconv.Atoi(s[p1+1:p2])
if err != nil {
    return date{}, false
}
d.m, err = strconv.Atoi(s[p1+1:p2])
if err != nil {
    return date{}, false
}
d.a, err = strconv.Atoi(s[p2+1:])
if err != nil {
    return date{}, false
}
if !dateOk(d) {
    return d, false
}
return d, true
```

}

- Scrivete le funzioni

dateOK (d data) bool

daysFromEpoch (d data) int
(1/1/1970)

dayOfWeek (d data) string

Lun, Mar, Mer, Gio, ...

[1/1/1970 giovedì]