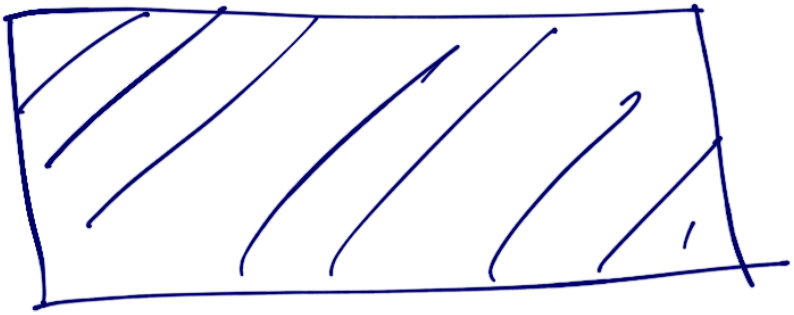
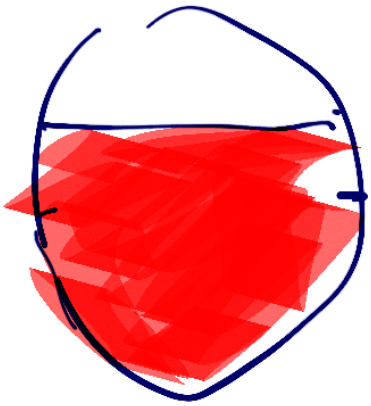
pippo.go

```
package main
import ... ]
func main() {
    
}
```

```
$> go build pippo.go
$> ./pippo
```

IMPORTAZIONE



"Batteries included"

```
import "fmt"
import "math"
import "math/random"
```

```
import (
    "fmt"
    "math"
    "math/random"
)
```

PACCHETTO fmt

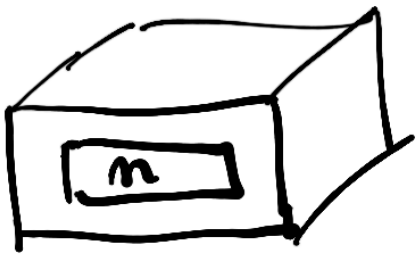
fmt = format

```
import "fmt"

...

func main () {
...
    fmt.Println("Quante volte? ");
    fmt.Println("Ciao")
    fmt.Print("Ciao")
```

VARIABILI



VARIABILE
- NOME ~ identif.
- TIPO
- SCOPE
- VALORE

```
var  n, i  int
```

var m, i int

m = 7

i = m + 3

SINTASSI

var $\{id_1\}, \{id_2\}, \ldots$ {tipo}

var (

{id}, {id}, ... {tipo}

{id}, {id}, ... {tipo}

)

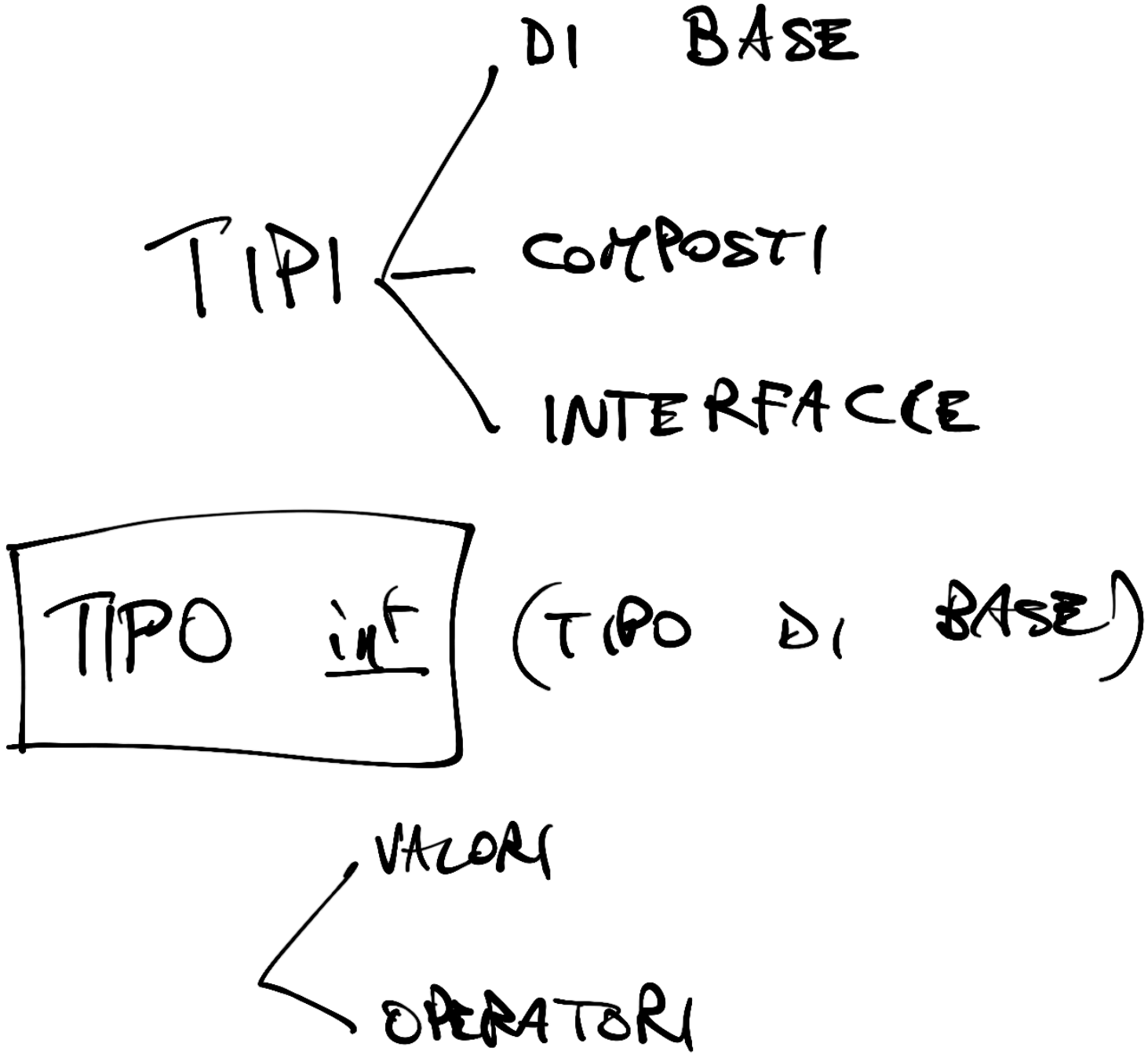
```
func main() {
    ...
    var pippo, i int
    for i=0; i<n; i++ {
        var pippo int
    }
}
```

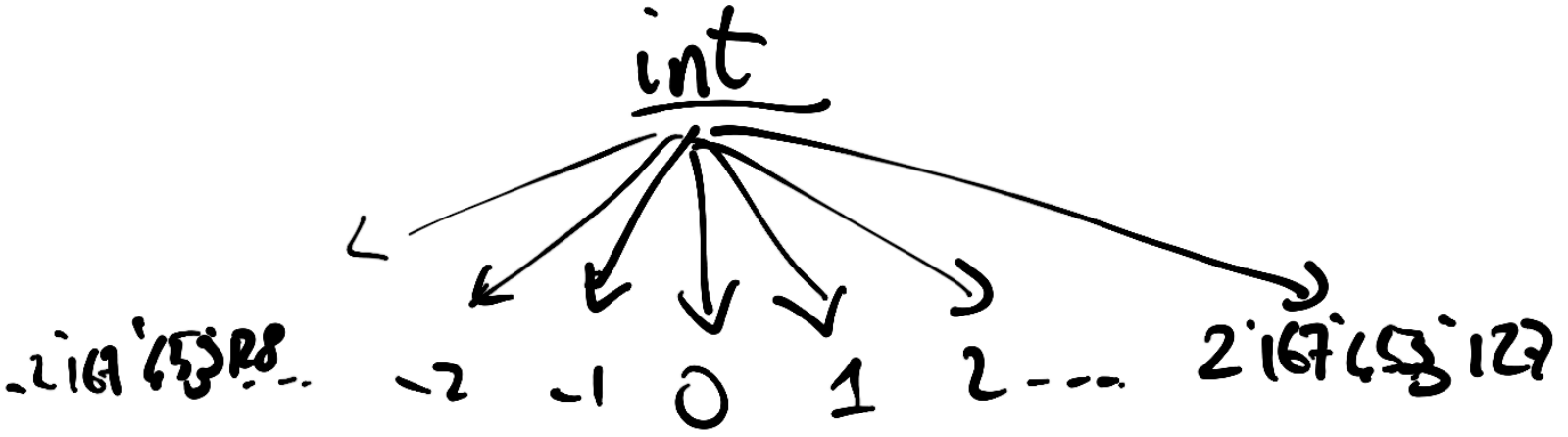
scope pippo

scope i

shadowing

2 locali uguali


DI BASE
TIPI
COMPOSTI
INTERFACCE
TIPO int (TIPO DI BASE)
VALORI
OPERATORI



---

var a, b, c, d int

fmt.Scan(&a)

ASSEGNAMENTO

var = espressione

Var a, b int.

a = b + 1

a = a + 1

a = a + 1

b = a * b + a * a + 5

a = (a + 1) * a + 1

a = b / 3 ← OVERLOADING DI /

a = b - 3

a = b % 5

a 4

b 14

Var a, b int

a = 17

b = a / 20

b = a % 20

b = a % 10

a = 1753

b = a % 10

b = (a / 10) % 10

var a, b int

a = 17

b = a % 2

a = 18

b = a % 2

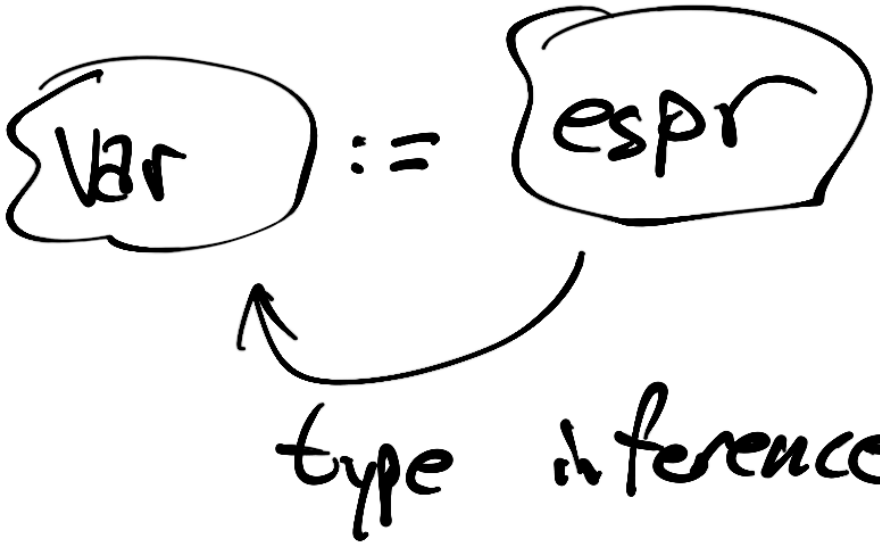
a % 2 = b

int	binari	unari
	+	+
	-	-
	*	
	/	
	%	

a = - b

a = +b

SHORT ASSIGNMENT



DICHIARAZIONE + ASSEGNAZIONE