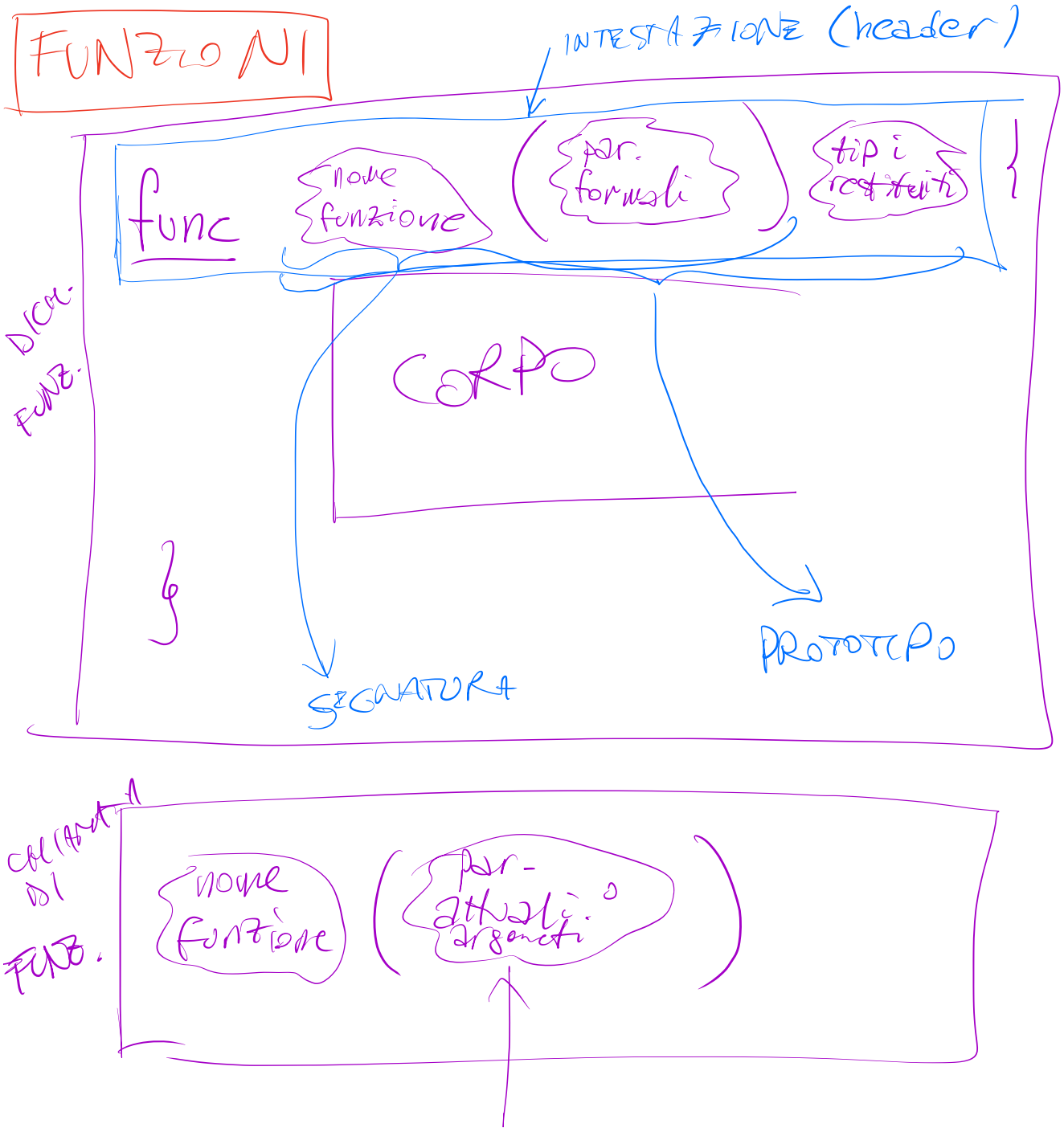
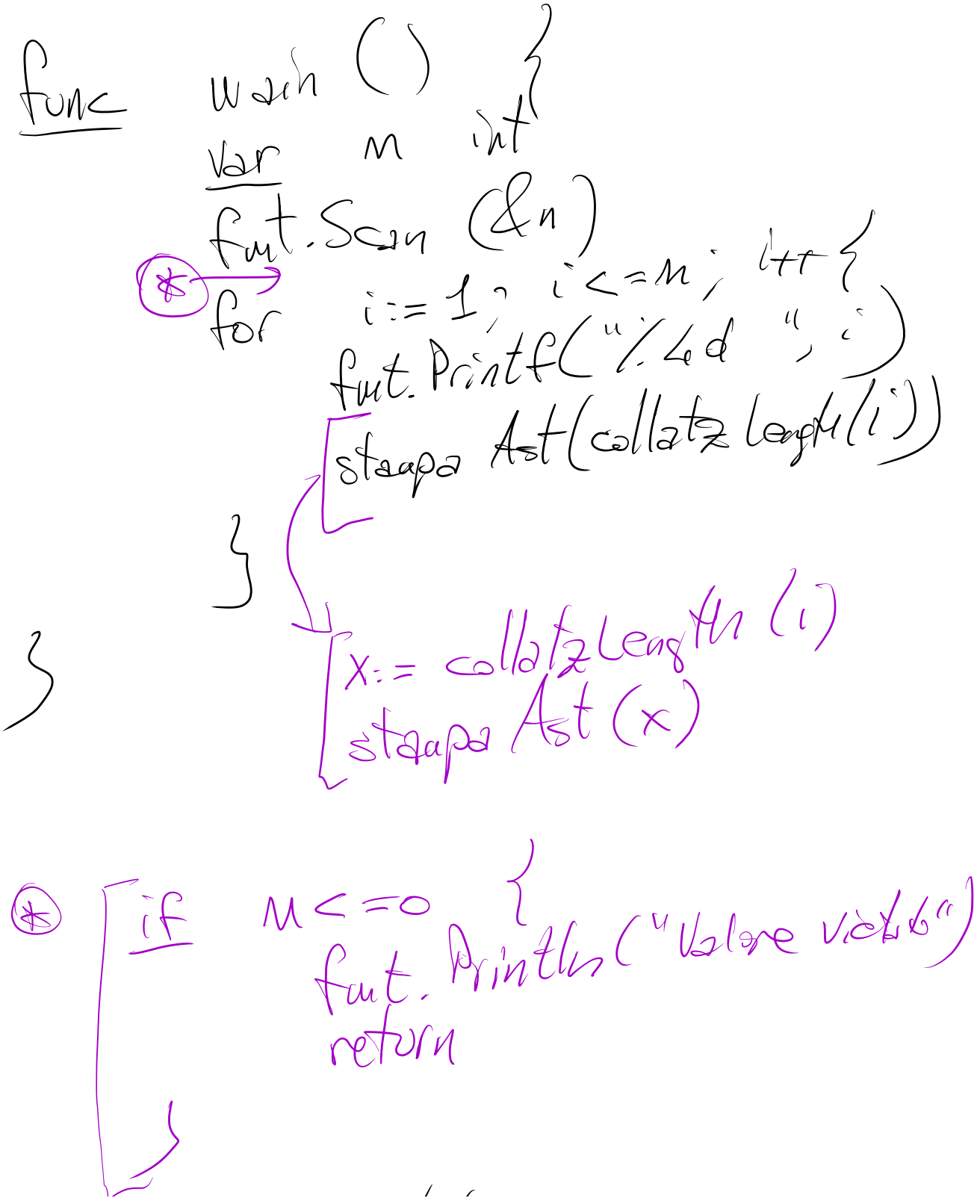

FUNZIONI
INTESTAZIONE (header)
func
nome funzione
par. formali
tipi restituiti
DICH. FUNZ.
CORPO
{
}
SIGNATURA
PROTOTIPO
CHIAMATA DI FUNZ.
nome funzione
par. attuali o argomenti

```
// nextCollatz(x) restituisce il
// numero che segue x
// nella sequenza di Collatz.
// x si assume >= 1.
func nextCollatz(n int) int {
    if n%2 == 0 {
        return n/2
    } else {
        return 3*n+1
    }
}

// collatzLength(x) restituisce
// la lunghezza della seq. di
// Collatz che parte da x >= 1
```

```
// e.g. Collatz Length (4) = 3
func collatzLength(x int) int {
    c := 1
    for x > 1 {
        x = nextCollatz(x)
        c++
    }
    return c
}

func stampAst(n int) {
    for i := 0; i < n; i++ {
        fmt.Print("*")
    }
    fmt.Println()
}
```


func main()
var n int
fmt.Scan(&n)
for i := 1; i <= n; i++ {
fmt.Printf("%4d", collatzLength(i))
}
}
x := collatzLength(i)
stampa Ast(x)
if n <= 0 {
fmt.Println("Valore vietato")
return
}

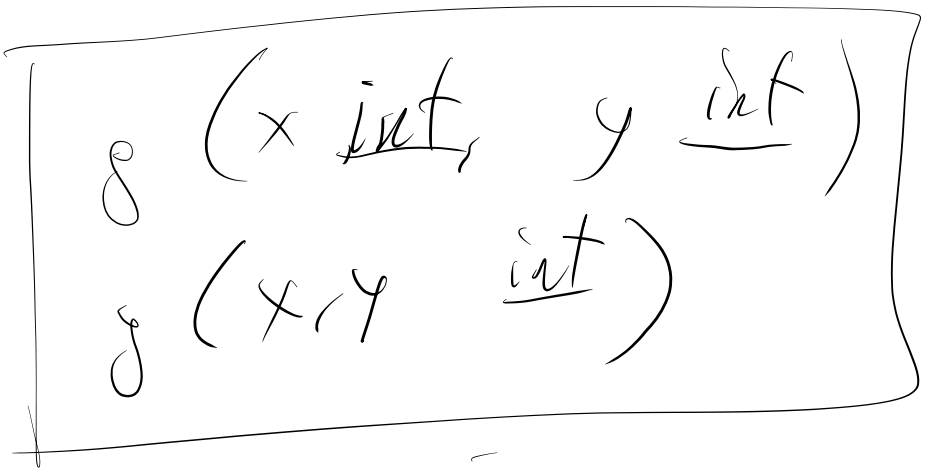
```
// Dato x, restituisce
// la parte intera e la
// parte frazionaria. Ad es.
// parti(17.34) restituisce 17
// e 0.34

func parti(x float64) (int, float64) {
    pi := int(x)
    pf := x - float64(pi)
    return pi, pf
}

return int(x), x - float64(int(x))
```

func f(x float64, y int)
string, bool {

- - -

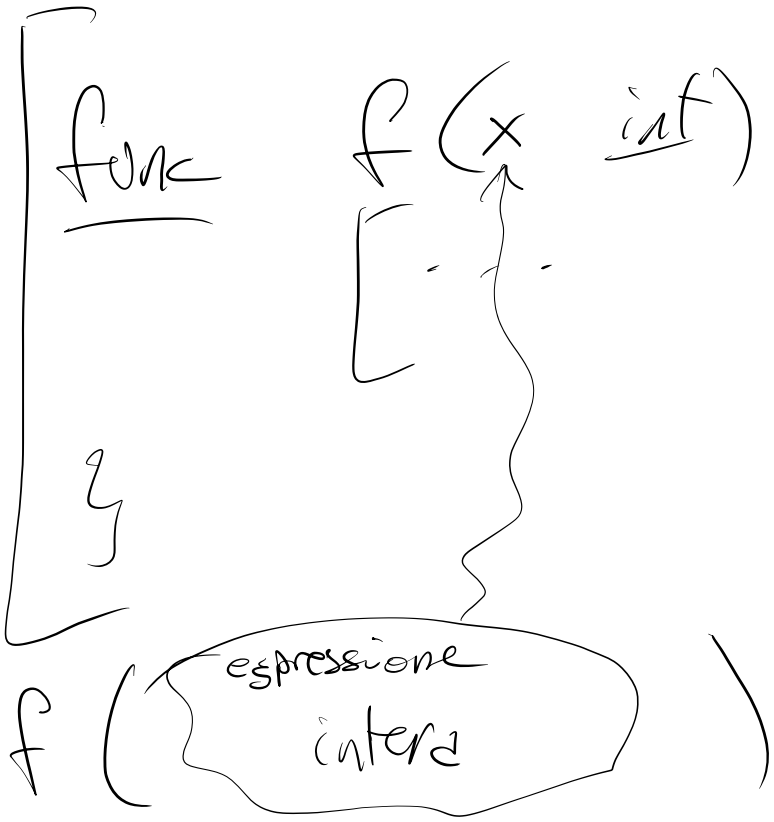
}

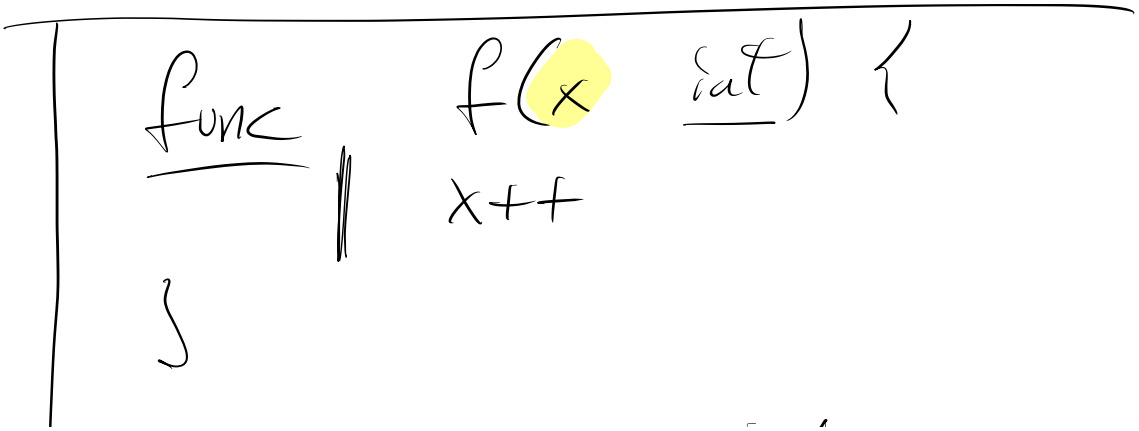
func

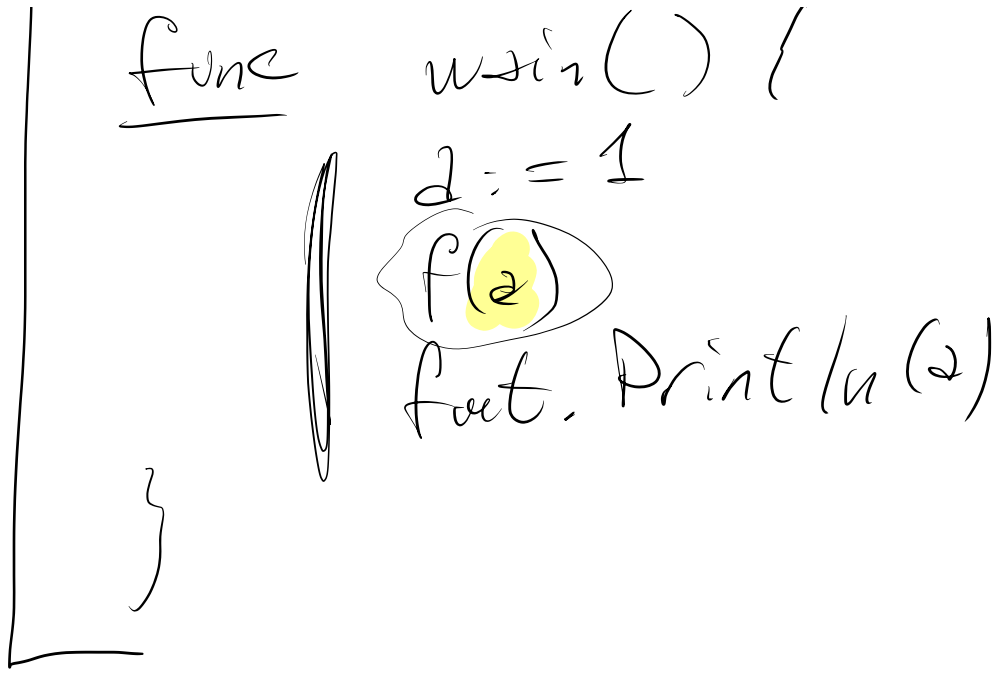


```
func main() {
var f float64
fmt.Scan(&f)
var x int
var y float64
x, y = math.Modf(f)
...
x, _ = math.Modf(f)
```

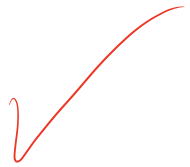
IN GO IL PASSAGGIO DI PARAMETRI AVVIENE PER VALORE



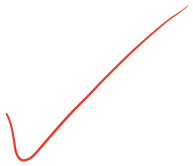



func main() {
d := 1
f(d)
fmt.Println(d)
}

```
var y float64
var x int

x, y = parti(f)
```



```
x, y := parti(f)
```

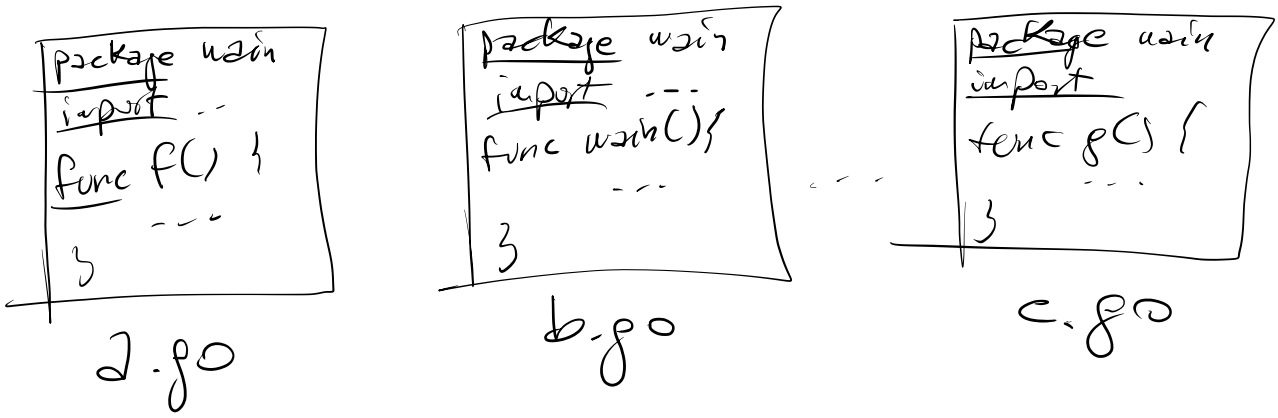


```
var x int

x, y := parti(f)
```

```
var x int
for i=0; i<n; i++ {
    var x int
    x,y := p2ti(float64(i),
}
```

GO

NO OVERLOADING



go build a.go b.go c.go

-o risultato

go run a.go b.go c.go

Primo di Mersenne

- Un numero intero si chiama primo di Mersenne sse è della forma $2^p - 1$ dove p è primo.

Voglio scrivere un programma che calcola l'n-esimo primo di Mersenne.

$$127 = 2^7 - 1$$

- func isPrime(x int) bool
- func isPowerOfTwo(x int) bool
- func expOfTwo(x int) int

```
func main() {
    var m, n int
    fmt.Scan(&n)
    c := 0
    for n := 1; ; n++ {
        if isPrime(n) &&
            isPowerOfTwo(n+1) {
```

```
p := exponentiation(n)
if isPrime(p) {
    c++
    if c == n {
        break
    }
}
}
}
fmt.Println(n)
}
```