STABILIRE SE UN NUMERO È PRIMO

```
var n, d int
fmt.Scan(&n)
for d = 2; d<n ; d++ {
    if n % d == 0 {
        break
    }
}
if d<n {
    fmt.Println("Composto")
} else {
```

else {

fwt.Println("Primo")

}

ISTRUZIONI break/continue

① break

② continue

① **break**: interrompe l'esecuzione del ciclo in cui è contenuta

for ( ) {

if ( )

```
        break
    }

for ... i<n ... {
    for ... j<m ... {
        for ... k<h ... {
            break
        }
        if k<h {
            break
        }
    }
    if j<m {
        break
    }
}
```

emersione rapida

② <u>continue</u>: forza la prossima esecuzione del ciclo

```
for i=1; i<n; i++{
  if i%10==0{
    continue
  }
}
```

- DATO n STAMPA I NUMERI PRIMI <= n

```
var n                    int
fmt.Scan(&n)
for x:=2; x<=n; x++ {
    if math.IsPrime(x) {
        fmt.Println(x)
    }
}
```

```
var n, d int
fmt.Scan(&n)
for x := 2; x <= n; x++ {
    for d := 2; d < x; d++ {
        if x % d == 0 {
            break
        }
    }
    if d == x {
        fmt.Println(x)
    }
}
```

- STAMPA I PRIMI N NUMERI PRIMI

```
var m, d, c int
fmt.Scan(&m)
for x:=2; c<m; x++ {
    for d=2; d<x; d++{
        if x%d ==0 {
            break
        }
    }
    if d==x {
        fmt.Println(x)
        c++
        if c==m {
            break
        }
    }
}
```

FUNZIONI

```
package main
import ...
func isPrime (n int) bool {
}
func main() {
    var n int
    fmt.Scan (&n)
    for x:=2; x<=n; x++ {
        if isPrime(x) {
            fmt.Println(x)
        }
    }
}
```

}

FUNZIONE

HEADER

func nome funzione (parametri formali) tipo rest. {

CORPO DELLA FUNZIONE

}

FUNZIONE CHE CALCOLA IL QUADRATO DI UN NUMERO INTERO

parametri formali

```
func sqr (a int) int {
    var ris int
    ris = a*a
    return ris
}

func main() {
    var n int
    fmt.Scan(&n)
    x := sqr(n)
    fmt.Println(x)
    x = sqr(5)
```

parametri attuali o argomenti

```
x := sqr(5)
fmt.Println(x)
y := sqr((x+1)*7)
fmt.Print(y)
```

~~z := sqr(x, y)~~

~~z := sqr()~~

~~z := sqr(5.0)~~

```
func ipotenusa(cat1, cat2 float64)
                          float64 {
    var x float64
    x = cat1*cat1 + cat2*cat2
    return math.Sqrt(x)
```

```
}

func main() {
    x := ipotenusa(7.0, 8.5)
    fmt.Println(x)
}

return math.Sqrt(cat1 * cat1
    + cat2 * cat2)
```

- SCRIVETE UNA FUNZIONE CHE DETERMINA SE UN INTERO È PARI

```
func isEven(x int) bool {
```

```
return x%2==0

}

func main() {
    var n int
    fmt.Scan(&n)
    if isEven(n) {
        fmt.Println("È pari")
    } else {
        fmt.Println("È dispari")
    }
    if isEven(n*n+7) && x>0{
```

}

ESERCIZI

- Scrivere una funzione che determina se un numero è divisore di un altro
- Scrivete una funzione che calcola quanti zeri compaiono nella scrittura di un numero. Usatela per calcolare quanti 0 dovete scrivere se scrivete tutti i numeri da 1 a 1000000.
- Scrivete una funzione che date le coordinate

di due punti sul piano cartesiano restituisce il coeff. angolare della retta passante.

BACK TO THE FUTURE

```
func isPrime(x int) bool {
    var d int
    for d=2; d<x; d++ {
        if x%d==0 {
            break
        }
    }
    if d==x {
        return true
    }
    else {
        return false
    }
```

```
}
}

func isPrime(x int) bool {
    var d int
    for d=2; d<x; d++ {
        if x%d==0 {
            break
        }
    }
    return d==x
}
```

```
func isPrime(x int) bool {
    var d int
    for d=2; d<x; d++ {
        if x%d==0 {
            return false
        }
    }
    return true
}

func main() {
    var M int
    fmt.Scan(&M)
```

```
for x:=2; x<n; x++{
    if isPrime(x){
        fat.Println(x)
    }
}
}
```

ESERCIZIO

- Un numero è primo di Mersenne se è primo ed è della forma $2^p - 1$ con p primo.

Scrivi una funz. che determina se un numero è primo di

Mersenne.

Usale per stampare i primi n primi di Mersenne.