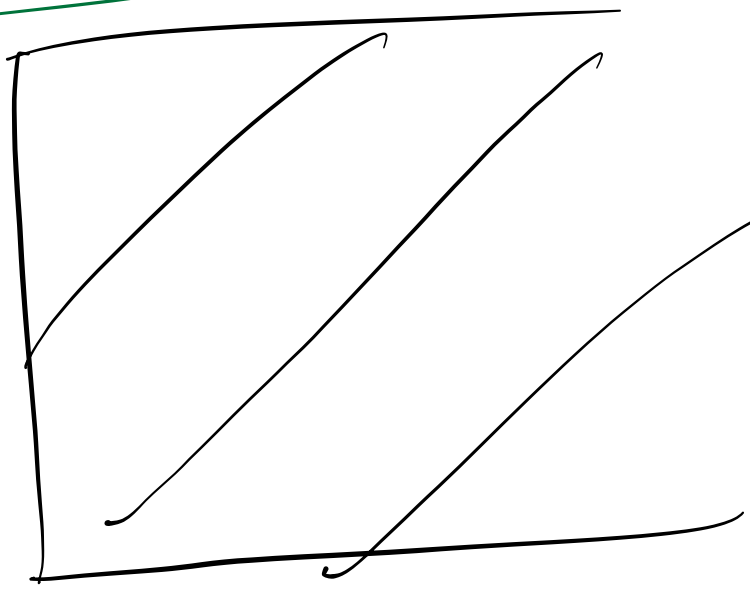
```
var n, stampati int
fmt.Scan(&n)
for d:=2; stampati<n; d++{
    if isPrime(d) {
        fmt.Println(d)
        stampati++
    }
}
```

primo attuale

package main

import "fmt"

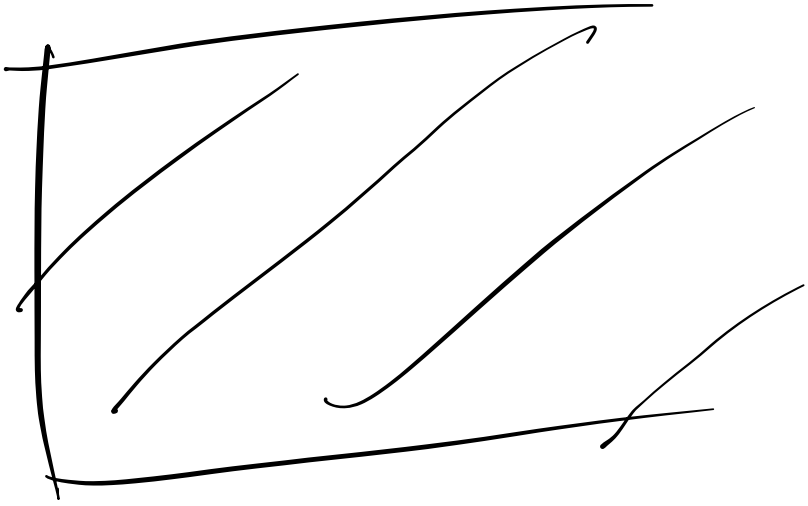
func isPrime(x int) bool {



corpo

}

func main() {



}

INTESTAZIONE

func (nome) (parametri formali) tipo ret.

```
/* Dato un intero x la funzione restituisce true o false a seconda che x sia primo o no */

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

return true

}

}

```
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 pippo(...) ... {
    for {
        ...
        if x*x >= 0 {
            return true
        } else {
            break
        }
        ...
    }
    return true // Can't happen
}
```

```
/* Dato n, restituisce il numero di
   primi <=n */
func countPrime(n int) int {
    var d, c int
    for d=2; d<=n; d++ {
        if isPrime(d) {
            c++
        }
    }
    return c
}
/* Restituisce n/ln n */
func asintotico(n int) float64 {
    return float64(n)/math.Log(
        float64(n))
}
```

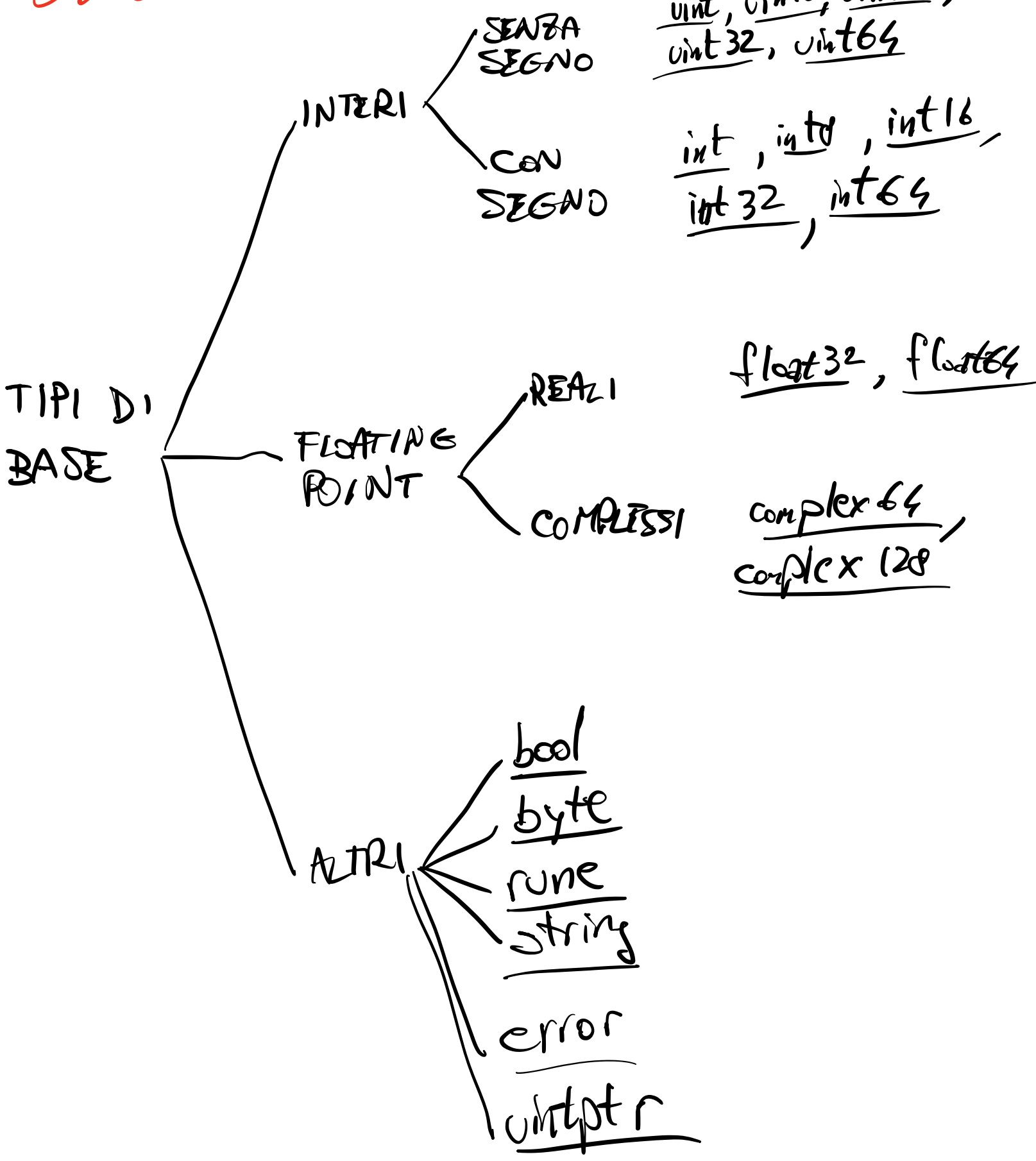
```
func main () {
    var n int
    n = 2
    for {
        x1 := float64(countPrime(n))
        x2 := assintotico(n)
        fmt.Println(n, x1/x2)
        n++
    }
}
```

```
func isEven(x int) bool {
    return x%2 == 0
}

func main() {
    ...
    for x := 2; x < 1000; x++ {
        if isEven(x) {
            fmt.Println(x)
        }
    }
    ...
}

isEven(x*x+1)
```

RAPPRESENTAZIONE DELL'INFORMAZIONE



TIPI INTERI

STRETTA SEGNO

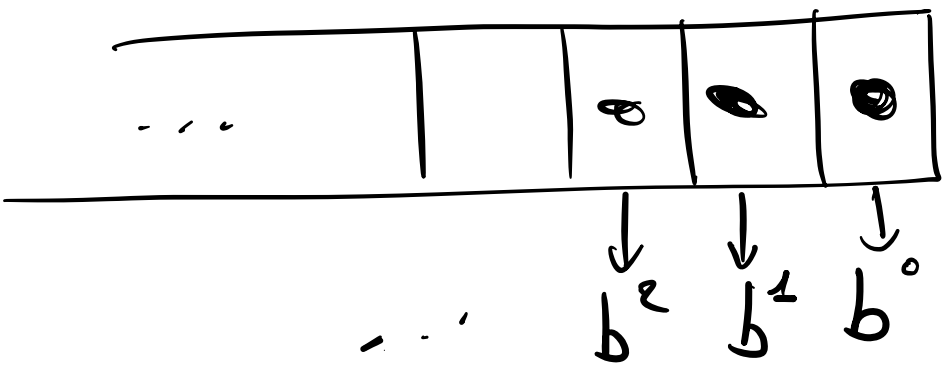
$\mathbb{N}$

CXXXIII

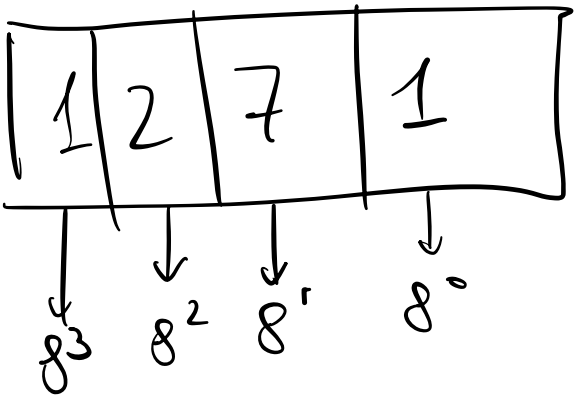
133

313

331



$$1271_8 = 697_{10}$$



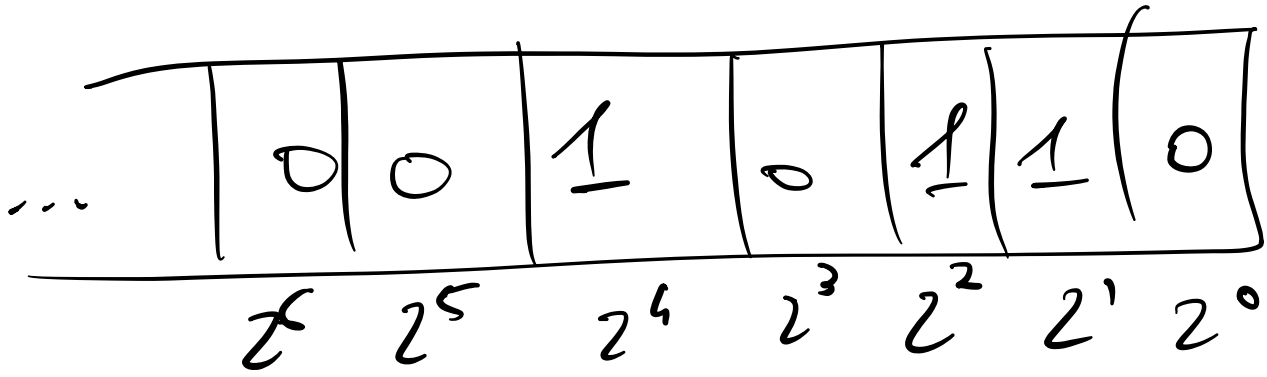
$1 \times 8^3 + 2 \times 8^2 + 7 \times 8^1 + 1 \times 8^0 =$

$= 512 + 128 + 56 + 1 =$

$= 697$

512
128
56
1

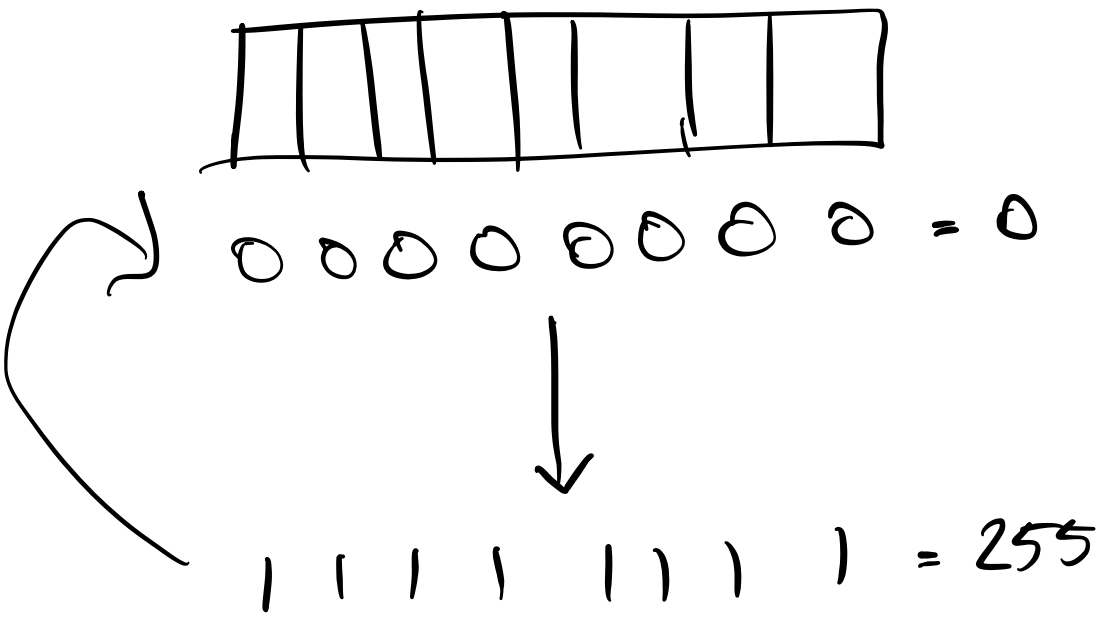
697



$10110_2 = 2^4 + 2^2 + 2^1 =$

$= 16 + 4 + 2 = 22$

var x uint8



TIPO	#BIT	#BYTE	RANGE
uint8	8	1	$0 \rightarrow 255$
uint16	16	2	$0 \rightarrow 65535$
uint32	32	4	$0 \rightarrow 4.294.967.295$
uint64	64	8	$0 \rightarrow \approx 1,84 \cdot 10^{19}$
uint	?		