# Appunti utili esame di programmazione

**Informazioni da ricordare**

- Dichiara costanti usando CONST (si può fare anche a livello di package)
- I caratteri non rune occupano più di un byte. Un carattere è byte se c > 127

**Metodi**

Sintassi:

```
func (receiverVar receiverType) methodName(args list) (return list)
```

Metodo String()

```
func (myVar myType) String() string {
        return someString
}
```

**Comandi di libreria utili**

rand.Intn(10) : Genera numeri casuali (in questo caso da o a 10)

Vecchio metodo:
import "math/rand"
rand.Seed(time.Now().Unix())
fmt.Println("Numero generato:", rand.Intn(100))

math.Abs(x-10) : Calcola il valore assoluto, si può utilizzare per fare confronti tra numeri float per vedere se la differenza tra i due è minore di un determinato numero evitando confronti tra float

math.Hypot(x, y) : Calcola x*x + y*y senza errori di overflow o generati da float

bufio.ScanWords : Modifica uno scanner per leggere le parole e non le righe

bufio.ScanRunes : Modifica uno scanner per leggere le rune e non le righe

time.Sleep(time.Duration(1) * time.Second) : Aspetta un secondo

strings.Repeat : Ripete più volte la stessa stringa

strings.Split : Divide una stringa dopo un carattere

strings.SplitAfter: Divide una stringa (carattere incluso)

strings.Fields : Prende una stringa con degli spazi e la trasforma in una

ParseFloat = string to float

**Funzioni**

**Scandire una stringa carattere per carattere:**

```
for i, c := range s{

}
```

i = indice stringa, c = carattere rune, s = stringa da scandire

**Passare da un numero decimale a binario:**

```
var n, x, N, i int
	fmt.Println("Programma che converte in decimale un numero binario inserito")
	fmt.Print("Inserisci un valore binario: ")
	fmt.Scan(&n)
	fmt.Print("Il numero ", n, "convertito in decimale è uguale a ")
	for n != 0{
		x = n % 10
		n/=10
		N+= x * int(math.Pow(2, float64(i)))
		i++
	}
	fmt.Println(N)
```

**Anno bisestile:**

```
func isLeap (a int) bool{
	return (a%4==0 && a%100!=0 || a%400 ==0)
}
```

**Tipi di stampe di una mappa:**

```
  // stampa dell'intera mappa
	fmt.Println(myMap)

  // stampa elemento per elemento, non importa in che ordine
	for key, value := range myMap {
		fmt.Println(key, value)
	}

  // stampa elemento per elemento, in ordine crescente di chiave, chiavi non presenti (con valore
zero) incluse
	for key := min; key <= max; key++ {
		fmt.Println(key, myMap[key])
	}

  // stampa elemento per elemento, in ordine crescente di chiave, chiavi non presenti escluse

	for key := min; key <= max; key++ {
		if value, ok := myMap[key]; ok {
			fmt.Println(key, value)
		}
	}
```

Cose da riguardare

substringing [:1] [1:]

fmt.Scanf("%c", &carattere) (I vari placeholder per scanf) e printf %v per float64
Funzioni unicode
Ordinamento e stampa mappe in orine alfabetico ecc
https://gobyexample.com/string-formatting SEGNA TUTTI
USO printf
USO SPRINTF GUARDA
GESTIONE FILE
GUARDA FUNZIONI RICORSIVE

CREAZIONE DI UN METODO

```
//Autore: Francesco Corrado

package main

import(
"fmt"
"os"
"strconv"
)

type Rettangolo struct{
base, altezza int
}

func (re Rettangolo) String() string{
var output string
if re.base == 0 || re.altezza == 0{
output = "rettangolo degenere"
}else{
for i := 0; i < re.altezza; i++{
for j := 0; j < re.base; j++{
output += "."
}
output += "\n"
}
}
return output
}

func main(){
var r Rettangolo
r.base, _ = strconv.Atoi(os.Args[1])
r.altezza, _ = strconv.Atoi(os.Args[2])
fmt.Println(r.String())
}
```

%q = strings