Laboratorio 10 - Package

1 Qual è l'output?

Si considerino le directory:

```
./
./rettangolo/
```

in cui sono sono presenti i file:

```
./area.go
./rettangolo/rettangolo.go
```

Nel file `rettangolo.go` è definito il seguente package `rettangolo`:

```go
package rettangolo

type Rettangolo struct {
    Base, Altezza float64
}

/* Restituisce una nuova istanza del tipo 'Rettangolo' inizializzata
in base ai valori dei parametri 'base' e 'altezza'. */
func NuovoRettangolo(base, altezza float64) *Rettangolo {
    return &Rettangolo{base, altezza}
}

/* Riceve in input un'instanza del tipo `Rettangolo` nel parametro `r` e
restituisce un valore `float64` pari all'area del rettangolo
rappresentato da `r`. */
func Area(r *Rettangolo) float64 {
    return r.Base * r.Altezza
}
```

Supponendo che l'utente inserisca da **riga di comando** i valori 10 e 12, cosa dovrebbe produrre in output il seguente programma scritto nel file `area.go`?

```go
package main

import (
    "./rettangolo"
    "fmt"
    "os"
```

```
    "strconv"
)

func main() {
    base, _ := strconv.ParseFloat(os.Args[1], 64)
    altezza, _ := strconv.ParseFloat(os.Args[2], 64)

    r := rettangolo.NuovoRettangolo(base, altezza)

    fmt.Println(rettangolo.Area(r))
}
```

2 Perimetro e Area di un triangolo

Scrivere un programma che:

- legga da **riga di comando** tre valori reali che corrispondono alle misure dei lati di un triangolo;
- stampi a video il valore del perimetro e dell'area del triangolo corrispondente ai tre valori reali letti.

Il programma deve utilizzare le funzionalità messe a disposizione da un package `triangolo` in cui è definito il tipo `Triangolo`:

```
type Triangolo struct {
    lato1, lato2, lato3 float64
}
```

e le seguenti funzioni:

- una funzione `NuovoTriangolo(l1, l2, l3 float64) (t *Triangolo, err bool)` che, se `l1+l2 > l3`, `l1+l3 > l2` e `l2+l3 > l1`, restituisce una nuova istanza del tipo `Triangolo` inizializzata in base ai valori dei parametri `l1`, `l1` e `l3` (nella variabile `t`) ed il valore `false` (nella variabile `err`); `nil` e `true` altrimenti;
- una funzione `Perimetro(t Triangolo) float64` che riceve in input un'instanza del tipo `Triangolo` nel parametro `t` e restituisce un valore `float64` pari al perimetro del triangolo rappresentato da `t`;
- una funzione `Area(t Triangolo) float64` che riceve in input un'instanza del tipo `Triangolo` nel parametro `t` e restituisce un valore `float64` pari all'area del triangolo rappresentato da `t`, calcolato utilizzando la formula di Erone:

```
p := (lato1 + lato2 + lato3) / 2
area := math.Sqrt(p * (p-lato1) * (p-lato2) * (p-lato3))
```

Esempio d'esecuzione:

```
$ go run area_e_perimetro.go 5 4 3
Perimetro del triangolo = 12
Area del triangolo = 6

$ go run area_e_perimetro.go 4 4 4
Perimetro del triangolo = 12
Area del triangolo = 6.928203230275509

$ go run area_e_perimetro.go 10 3 5
Errore: Impossibile creare un triangolo in base alle misure specificate per i tre lati!
```

3 Triangoli casuali

Si estenda il package `triangolo` definito relativamente all'Esercizio 2 (Laboratorio 9 - Package) implementando la funzione:

- `String(t Triangolo) string` che riceve in input un'instanza del tipo `Triangolo` nel parametro `t` e restituisce un valore `string` che corrisponde alla rappresentazione `string` di `t` nel formato `Triangolo con lati L1, L2 e L3.`, dove `L1`, `L2` ed `L3` sono i valori `string` corrispondenti ai valori dei campi `lato1`, `lato2` e `lato3` di `t`.

Utilizzando le funzionalità messe a disposizione dal package `triangolo`, scrivere un programma che:

- legga da **riga di comando** un numero intero `n`;
- generi in maniera casuale `n` triple di valori reali compresi tra `10` e `1000`; i valori `l1`, `l2`, `l3` di ciascuna tripla corrispondono alle misure dei lati di un ipotetico triangolo;
- stampi a video la rappresentazione `string` del triangolo con area più grande tra quelli corrispondenti alle triple di valori reali generate;
- stampi a video la rappresentazione `string` del triangolo con perimetro più piccolo tra quelli corrispondenti alle triple di valori reali generate.

Oltre alla funzione `main()`, devono essere definite ed utilizzate almeno le seguenti funzioni:

- una funzione `GeneraTriangoli(n int) (tN []triangolo.Triangolo)` che riceve in input un valore `int` nel parametro `n` e restituisce un valore `[]triangolo.Triangolo` nella variabile `tN` in cui sono memorizzate le istanze del tipo `triangolo.Triangolo` inizializzate in base alle `n` triple di valori reali generate in maniera casuale (poiché la funzione `triangolo.NuovoTriangolo(l1, l2, l3 float64) (t *triangolo.Triangolo, err bool)` restituisce i valori `nil` e `true` nel caso in cui `l1+l2 <= l3`, `l1+l3 <= l2` o `l2+l3 <= l1`, potrebbe succedere che `len(tN) < n`).

Esempio d'esecuzione:

```
$ go run triangoli_casuali.go 10
Triangolo con area maggiore = Triangolo con lati 844.728464, 872.971432 e 644.031285
Triangolo con perimetro minore = Triangolo con lati 587.135063, 363.401214 e 612.413022
```