Laboratorio 10 - Package

1 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 error)` 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 `nil` (nella variabile `err`); `nil` e un errore 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))
  ```

Suggerimento: per creare una istanza di tipo `error` potete utilizzare la funzione `errors.New()` del package `errors`. Usate `go doc errors.New` per vederne il corretto funzionamento.

Esempio d'esecuzione:

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

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

$ go run area_e_perimetro.go 10 3 5
Errore: impossibile creare un triangolo con le misure specificate
```

2 Triangoli casuali

Si estenda il package `triangolo` definito relativamente all'esercizio **1 Perimetro e Area di un triangolo** 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 ai campi `lato1`, `lato2` e `lato3` di `t`. La conversione in `string` dei valori dei tre lati deve essere effettuata utilizzando due cifre decimali.

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 di tipo `*triangolo.Triangolo` valide (generate senza restituire un errore).

Esempio d'esecuzione:

```
$ go run triangoli_casuali.go 10
Triangolo con area maggiore = Triangolo con lati 716.05, 867.05 e 976.47
Triangolo con perimetro minore = Triangolo con lati 590.89, 266.73 e 559.86
```