Laboratorio 3 - Iterazione

1 Qual è l'output?

Supponendo che l'utente inserisca da **standard input** `10`, qual è l'output di questo programma? E se inserisse `11`? Che cosa fa questo programma?

```
package main

import "fmt"

func main() {
    var n int
    fmt.Print("Inserisci un numero: ")
    fmt.Scan(&n)

    var i int
    for i = 0; i <= n; i += 2 {
        fmt.Print(i, " ")
    }
    fmt.Println()
}
```

2 Qual è l'output?

Supponendo che l'utente inserisca da **standard input** `36`, qual è l'output di questo programma? E se inserisse `32`? Che cosa fa questo programma? Quali sono le differenze con il programma dell'esercizio precedente?

```
package main

import "fmt"

func main() {

    var n int

    fmt.Print("Inserisci un numero: ")
    fmt.Scan(&n)

    var i int
    for i = 1; i <= n; i *= 2 {
        fmt.Print(i, " ")
    }

    fmt.Println()
```

```
}
```

3 Trova gli errori

Questo programma è errato e non produce l'output descritto nel commento del package. Correggilo (3 errori: 1 di sintassi, 2 di logica del programma).

```
/*
Il programma stampa una sequenza di numeri

Dato un numero n inserito da tastiera, il programma stampa tutti i numeri
compresi nell'intervallo che va da 1 a n (estremi inclusi).
La sequenza è prodotta su un'unica riga di testo in cui ciascun numero è separato
dal precedente da uno spazio.
*/
package main

import (
    "fmt"
)

func main() {

    var n int

    fmt.Print("Inserisci un numero: ")
    fmt.Scan(&n)

    var i int;
    for i = 1; i < n {
        fmt.Print(i)
        i++
    }

    fmt.Println()

}
```

4 Tabellina

Scrivere un programma che, dopo aver richiesto all'utente di inserire da **standard input** un numero intero `n`, stampi a video la corrispondente tabellina (moltiplicando `n` per i numeri naturali da 1 a 10) come mostrato nell'**Esempio d'esecuzione**.

Esempio d'esecuzione:

```
$ go run tabelline.go
Inserisci un numero: 9
1 x 9 = 9
2 x 9 = 18
3 x 9 = 27
4 x 9 = 36
5 x 9 = 45
6 x 9 = 54
7 x 9 = 63
8 x 9 = 72
9 x 9 = 81
10 x 9 = 90
```

5 Somma di numeri in intervallo

Scrivere un programma che legga da **standard input** due numeri interi `a` e `b` e stampi a video la somma dei numeri pari compresi tra `a` e `b` (estremi esclusi).

Esempio d'esecuzione:

```
$ go run sommaintervallo.go
1 9
Somma = 20
```

6 Operazioni con un numero variabile di valori

Scrivere un programma che, dopo aver letto da **standard input** un numero intero `n`, chiede all'utente di inserire `n` numeri interi (sempre da **standard input**).

Dopo aver letto gli `n` numeri interi, il programma deve stampare: * la somma degli `n` numeri letti; * il minimo tra i numeri letti; * il massimo tra i numeri letti; * il numero di interi letti strettamente positivi (maggiori di 0), strettamente negativi (minori di 0), e nulli.

Esempio d'esecuzione:

```
$ go run nnumeri.go
9
Inserisci 9 numeri:
1 -2 3 -4 5 -6 7 -8 9
somma = 5
valore minimo = -8
valore massimo = 9
interi > 0 = 5
interi < 0 = 4
interi = 0 = 0
```

7 Media

Scrivere un programma che legga da **standard input** una sequenza di numeri reali strettamente positivi (un numero è strettamente positivo se è maggiore di 0; se un numero è minore o uguale 0 non è strettamente positivo). La lettura termina quando viene letto un numero minore o uguale a 0.

Il programma deve stampare a video il risultato della media aritmetica dei valori inseriti.

Esempio d'esecuzione:

```
$ go run medie.go
Inserisci una sequenza di numeri (interrompi con numero<=0): 4 6 8 0
Media aritmetica: 6

$ go run medie.go
Inserisci una sequenza di numeri (interrompi con numero<=0): 3 5 2 6 1 -1
Media aritmetica: 3.4
```

8 Indovina il numero

Scrivere un programma che:

1) Genera in modo casuale un numero intero compreso nell'intervallo che va da 1 a 100, estremi inclusi (sia `numeroGenerato` la variabile intera in cui viene memorizzato il numero generato, come indicato nella consegna deve valere 1<= `numeroGenerato` <= 100).
2) Chiede iterativamente all'utente di inserire da **standard input** un numero intero; ad ogni iterazione il programma controlla se il numero inserito è uguale al numero memorizzato in `numeroGenerato`:

- se sono uguali, il programma termina;
- se sono diversi, il programma fornisce un suggerimento specificando se il numero inserito è più alto o più basso di quello memorizzato in `numeroGenerato`.

L'output stampato a video dal programma deve essere quello riportato nell'**Esempio d'esecuzione** (eccezion fatta per i numeri inseriti dall'utente).

Suggerimento: per generare in modo casuale un numero intero, potete utilizzare le funzioni del package `math/rand` come mostrato nel seguente frammento di codice:

```
/* inizializzazione del generatore di numeri casuali */
rand.Seed(int64(time.Now().Nanosecond()))
/* generazione di un numero casuale compreso nell'intervallo
   che va da 0 a 99 (estremi inclusi) */
var numeroGenerato int = rand.Intn(100)
```

Esempio d'esecuzione:

```
$ go run indovina.go
Tentativo n° 1: 50
Troppo basso! Riprova!
Tentativo n° 2: 75
Troppo alto! Riprova!
Tentativo n° 3: 63
Troppo basso! Riprova!
Tentativo n° 4: 68
Troppo alto! Riprova!
Tentativo n° 5: 66
Hai indovinato in 5 tentativi!
```

9 Fizz Buzz

Scrivere un programma che legga da **standard input** un intero `n`. Il programma deve stampare a video la sequenza di numeri che vanno da 1 a `n` come mostrato nell'**Esempio d'esecuzione**. In particolare: * ogni numero divisibile per 3 deve essere rimpiazzato dalla parola `"Fizz"`; * ogni numero divisibile per 5 deve essere rimpiazzato dalla parola `"Buzz"`; * ogni numero divisibile sia per 3 sia per 5 deve essere sostituito da `"Fizz Buzz"`.

Esempio d'esecuzione:

```
$ go run fizzbuzz.go
6
1 2 Fizz 4 Buzz Fizz

$ go run fizzbuzz.go
20
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19 Buzz
```

10 Divisori

Scrivere un programma che legga da **standard input** un numero intero `n` e stampi a video i **divisori propri** del numero letto, ovvero tutti i suoi divisori escluso il numero stesso. Ad esempio, i **divisori** del numero 12 sono: 1, 2, 3, 4, 6, 12; quindi i **divisori propri** di 12 sono: 1, 2, 3, 4, 6.

Esempio d'esecuzione:

```
$ go run divisori.go
Inserisci numero: 6
Divisori di 6: 1 2 3

$ go run divisori.go
```

```
Inserisci numero: 10
Divisori di 10: 1 2 5
```

11 Operazioni

Create un programma che legge da **standard input** due numeri interi, che chiameremo `x` e `y`. Letti i due numeri, il programma stampa: * il maggiore tra `x` e `y` * il minore tra `x` e `y` * il risultato della somma tra `x` e `y` * il risultato della differenza tra il maggiore e il minore * il risultato della divisione `x/y` * il risultato del prodotto `x*y` * il valore medio tra `x` e `y` * il risultato di `x` elevato alla `y` (utilizzando sia un ciclo `for` sia la funzione `math.Pow`)

Esempio d'esecuzione:

```
$ go run operazioni.go
Inserisci due numeri interi:
2 4
Maggiore: 4
Minore: 2
Somma: 6
Differenza: 2
Prodotto: 8
Divisione: 0.5
Valore medio: 3
Potenza (ciclo for): 16
Potenza (math.Pow): 16
```

12 Sequenza crescente/decrescente

Il programma legge da tastiera una serie di numeri > -1 e dopo ogni lettura stampa "crescente" se il nuovo valore è maggiore o uguale al precedente e "dcrescente" altrimenti. Il programma si ferma quando il numero in input è <= -1. Utilizzare fmt.Scan per la lettura.

```
$ go run sequenza.go
inserisci una sequenza di numeri:

1 3 2 5 6 3 1 -1

crescente
crescente
decrescente
crescente
crescente
decrescente
decrescente
```

13 Fibonacci

Create un programma che legge da **standard input** un intero positivo `n` e stampi i primi `n` elementi della successione di Fibonacci. La successione di Fibonacci (detta anche successione aurea) è una successione di numeri interi in cui ciascun numero è la somma dei due precedenti.

```
$ go run fibonacci.go
Inserisci un numero intero:
4

1 1 2 3

$ go run fibonacci.go
Inserisci un numero intero:
5

1 1 2 3 5

$ go run fibonacci.go
Inserisci un numero intero:
12

1 1 2 3 5 8 13 21 34 55 89 144
```