Testing

Tipi di errore

Errori formali

Il programma viola alcune regole del linguaggio. Tipicamente segnalati dal compilatore o dall'interprete.
Esempio:

```
var a itn
fmt.Prinln(a)
```

Errori logici

Il programma è scritto utilizzando il linguaggio in modo corretto ma non produce l'output atteso.
Esempio:

```
for i = 0; i < 10; i-- {
  fmt.Println(i)
}
```

Debugging

Attività di ricerca e correzione degli errori (**bug**) presenti all'interno del programma.

Come cercare i bug?

- utilizzare messaggi di log
- scomporre il programma in unità indipendenti
- utilizzo di debugger
- testing

Testing

Attività che prevede di eseguire il programma (o parti del programma) molteplici volte, sottoponendo ad ogni esecuzione input diversi. Gli input sono pensati appositamente per cercare gli errori.

L'attività di testing **non** può garantire l'assenza di errori.

Livelli di testing

- unit testing: test di singole entità del programma che possono essere trattate in modo indipendente (i.e.: funzioni).
- integration testing: test sulla cooperazione tra entità diverse ma che interagiscono tra di loro.
- system testing: test del funzionamento dell'intero programma all'interno di un ambiente il più simile possibile all'ambiente designato per l'esecuzione del programma stesso.

Tipi di testing

- **funzionali**: verificano che il programma risponda in modo corretto agli input fornendo l'output atteso.
- **non funzionali**: verificano che il programma rispetti delle prorietà collaterali (e.g.: tempi di calcolo, consumo di memoria, ...)

Testing - esempio 1

Problema

Scrivere un programma che legga da **riga di comando** un numero intero positivo e stampi a video se tale numero è primo. Se il valore inserito non è un numero intero o è un numero intero non positivo, il programma stampa un messaggio d'errore.

Unit di test

Possibili unit di test:

- il valore inserito è un numero intero positivo primo
- il valore inserito è un numero intero positivo non primo
- il valore inserito è 1 (1 non è un numero primo)
- il valore inserito è un numero intero nullo
- il valore inserito è un numero intero negativo
- il valore inserito è un numero reale
- il valore inserito non è un numero

Testing - esempio 2

Problema

Scrivere un programma che legga da **standard input** un testo disposto su più righe e stampi a video il numero di vocali presenti nel testo.

Unit di test

Possibili unit di test:

- non viene inserito alcun testo
- il testo inserito ha una sola riga
- il testo inserito è disposto su più righe
- il testo inserito non ha vocali
- il testo inserito ha solo vocali
- il testo inserito contiene anche spazi
- il testo inserito contiene caratteri non ASCII

Test in Go (1)

Il package `testing` mette a disposizione degli strumenti per il testing.

```
import "testing"
```

Lo strumento `go test` esegue tutte le funzioni aventi prefisso `Test` presenti nei file con suffisso `_test.go` nella cartella corrente.

```
func TestFunzione(t *testing.T) {
    // codice della unit di test
}
```

Test in Go (2)

Ogni funzione con prefisso `Test` è una unit di test. Una unit di test fallisce se chiama una funzione `t.Error()` o `t.Errorf()`.

```
$ go test
--- FAIL: ...
```

Se nessuna unit fallisce, la fase di testing è superata.

```
$ go test
PASS
```