Corso di Laurea Ingegneria Informatica

Fondamenti di Informatica 1

Dispensa 10

Correttezza

A. Miola

Novembre 2007

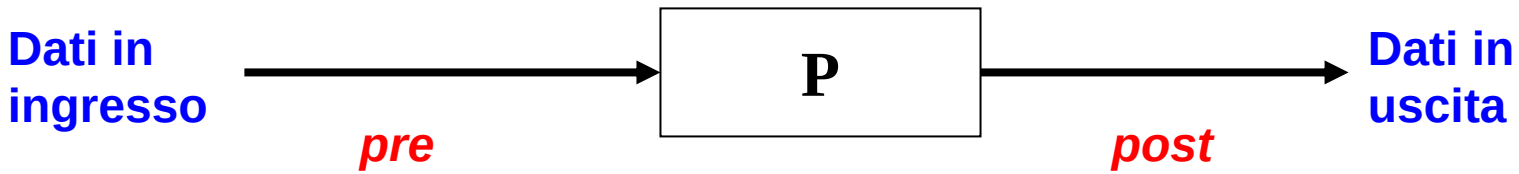
Contenuti

- Introduzione alla correttezza
- Correttezza dei metodi
  - specifica di un metodo
  - correttezza di un metodo
  - correttezza e responsabilità
- Verifica di correttezza
  - test a scatola nera
  - test a scatola trasparente
  - metodi di test
- Test di un insieme di metodi

Correttezza

- **La correttezza è una delle qualità fondamentali dei prodotti software**
  - **detto in modo molto preliminare e generico (come lo direbbe un uomo della strada) un prodotto software è "corretto" se consente le operazioni per cui è stato pensato**
- **Questa dispensa presenta la nozione di correttezza dei programmi, con particolare riferimento alla programmazione orientata agli oggetti, studiando gli aspetti e gli strumenti che permettono di descrivere e verificare la correttezza del software**

Esecuzione di un programma



- Un programma P riceve un insieme di **dati in ingresso** che devono soddisfare alcune condizioni, cioè hanno delle proprietà, che indichiamo con ***pre*-condizioni**
- Un programma P esegue le operazioni previste, in modo non ambiguo e in un tempo finito, cioè **termina**
- Un programma P fornisce un insieme di **dati in uscita** che devono soddisfare alcune condizioni, cioè hanno delle proprietà, che indichiamo con ***post*-condizioni**

Specifica di un programma

- L'espressione <*pre*, P, *post*> rappresenta la specifica del programma P
- La specifica di un programma P si dice soddisfatta se vale la seguente proprietà:

  Per ogni insieme di dati che soddisfano la pre-condizione *pre*, se il programma P termina quando viene eseguito su tali dati, allora i dati di uscita prodotti dal programma soddisfano la post-condizione *post*
- DEFINIZIONE - Un programma P è parzialmente corretto rispetto alla pre-condizione *pre* e alla post-condizione *post*, se la specifica <*pre*, P, *post*> è soddisfatta.

Esempio

- Supponiamo di utilizzare un programma P calcolare il massimo comun divisore m di due numeri x e y interi
- Il programma P può essere utilizzato ***sse*** i due numeri x e y sono interi e non entrambi nulli e conseguentemente la ***pre*-condizione** per P è

(x e y sono interi non negativi) and ((x > 0) or (y > 0))

- Il risultato calcolato da P, m = MCD(x, y), deve soddisfare la seguente ***post*-condizione**

(m divide x) and (m divide y) and
(ogni numero intero che divide sia x che y divide anche m)

Definizione

- **Un programma P è corretto rispetto alla pre-condizione *pre* e alla post-condizione *post* se è parzialmente corretto e se esso termina ogni volta che viene eseguito su dati di ingresso che soddisfano la pre-condizione *pre***

. . . OPPURE . . .

- **Un programma P è corretto rispetto alla specifica <*pre*, P, *post*> se per ogni insieme di dati di ingresso che soddisfa la pre-condizione *pre* il programma P termina e fornisce come risultato un valore che soddisfa la post-condizione *post***

Dimostrazione della correttezza . . .

- **In base alla definizione precedente la dimostrazione della correttezza di un programma consiste dei seguenti passi:**

1. **si determina la pre-condizione *pre* e la post-condizione *post*;**
2. **si verifica se la specifica è soddisfatta;**
3. **si stabilisce se il programma termina per ogni possibile insieme di dati di ingresso che soddisfa la pre-condizione.**

. . . Dimostrazione della correttezza

- Pur essendo possibile la dimostrazione formale e rigorosa della correttezza di un programma, bisogna considerare che tali prove risultano molto complesse e lunghe
- Per questa ragione si segue un approccio pragmatico che viene chiamato **metodo sperimentale di verifica della correttezza** di un programma o **test** di correttezza

Test di un programma . . .

- **Il test di un programma consiste dei seguenti passi:**

1. **si determina un insieme di dati di input I;**
2. **si esegue il programma con i dati I;**
3. **si verificano i risultati ottenuti:**

   3.1 se l'esecuzione del programma non termina in un lasso di tempo ragionevole o se i risultati ottenuti dalla esecuzione non sono quelli attesi, allora il programma non è corretto

   3.2 se non sono stati rilevati errori, bisogna stabilire se è necessario effettuare altre prove; in questo caso si ritorna al passo 1, altrimenti il test termina

. . . Test di un programma . . .

- Generalmente il test di un programma non permette di stabilirne la correttezza, a meno che non vengano provate tutte le possibili scelte di dati di ingresso; infatti, se eseguiamo il programma con un sottoinsieme dei possibili input non possiamo escludere che esista un insieme di dati, che non è stato provato, e che evidenzia un errore

. . . Test di un programma . . .

- Chiaramente, la prova di un programma con tutti i possibili dati di ingresso è impossibile nella maggioranza dei casi
- Ad esempio, l'insieme dei dati di ingresso di un programma che calcola il prodotto di due interi compresi tra -100000000 e +100000000 è composto da circa 40 milioni di miliardi di elementi. Se ciascuna prova richiede un microsecondo (0.000001 secondi) allora sono necessari 40 miliardi di secondi, circa 40000 giorni cioè circa 110 anni

. . . Test di un programma

- Possiamo pertanto concludere che, anche se generalmente il test di un programma non permette di dimostrarne la correttezza, esso è lo strumento più usato nella pratica della programmazione; infatti, se effettuiamo molte prove e non rileviamo errori, possiamo ragionevolmente supporre che la probabilità che il programma contenga errori sia molto piccola

Correttezza nella programmazione orientata ad oggetti

- In generale, è possibile ricondurre la verifica della correttezza di una classe o di una applicazione alla verifica del comportamento di un oggetto cioè dei suoi metodi
  - facciamo delle ipotesi semplificative
    - oggetti classe
    - senza stato
    - comportamento senza effetti collaterali – solo calcolo di funzioni
- La correttezza di un metodo è legata alla specifica del metodo – che è un particolare programma

Alcuni termini rilevanti

- **Malfunzionamento (o comportamento errato) di un oggetto**
  - una discrepanza tra l'effettivo comportamento esterno dell'oggetto e quello corretto
- **Oggetto non corretto (o errato)**
  - durante il suo uso possono verificarsi malfunzionamenti
  - un oggetto è **corretto** se durante il suo uso **non possono mai verificarsi** malfunzionamenti
- **Errore (o difetto)**
  - è la **causa di un malfunzionamento**
  - un malfunzionamento è una manifestazione di un errore

Verifica di correttezza

- Le tecniche di verifica di correttezza prevedono solitamente l'uso dell'oggetto da verificare, cercando di provocare eventuali malfunzionamenti e comportamenti non corretti
  - il verificarsi di un malfunzionamento ci garantisce la presenza di errori nell'oggetto
  - l'assenza di malfunzionamenti certifica la correttezza dell'oggetto solo se l'oggetto è stato utilizzato in tutti i possibili modi
    - in pratica, però, la verifica di correttezza di un oggetto viene in genere basata su un numero limitato di prove
- La verifica di correttezza di un oggetto può semplicemente confermare la presenza di errori, e mai la loro assenza

Individuazione e correzioni degli errori

- **Individuazione degli errori**
  - attività di ricerca degli errori
    - la verifica di correttezza non dice quanti e quali sono
  - tipologie di errori
    - errori di uso del linguaggio di programmazione
    - errori di codifica
    - errori nella realizzazione dell'algoritmo
    - errori nella scelta dell'algoritmo
  - è problematico solo trovare gli errori non riconosciuti dal compilatore
- **la correzione di un errore è tanto più costosa quanto più tardi l'errore viene rilevato e corretto rispetto al momento in cui è stato introdotto**

Specifica di un metodo

- La nozione di specifica di un metodo formalizza l'idea di comportamento per cui il metodo è stato pensato
- La specifica di un metodo corrisponde alla specifica del problema che il metodo deve risolvere
- La specifica di un metodo è parte della sua documentazione

Scelta dei dati di ingresso per un test

- **Con quali dati effettuare il test di un metodo?**
  - ci sono varie tecniche di verifica, che si differenziano sulla scelta dell'insieme dei dati di test su cui basare la verifica
- **Due approcci principali**
  - test a scatola nera
    - l'insieme dei dati di ingresso viene scelto solo in riferimento alla specifica del metodo, senza far riferimento ai dettagli realizzativi del metodo
  - test a scatola trasparente
    - l'insieme dei dati di ingresso viene scelto anche in riferimento ai dettagli realizzativi del metodo

Evitare scelte casuali dei dati di test

- È importante premettere che il test di un programma non consiste semplicemente nell'eseguire il programma una o più volte con dati scelti in modo casuale, ma è un'attività che deve essere condotta e pianificata durante tutta la fase di progettazione
- L'esempio seguente mostra come una scelta casuale dei dati di prova non risulta adeguata

Esempio . . .

- Siano date due stringhe a e b, di al più 20 caratteri, di lunghezza pari a lunA e lunB rispettivamente
- Si consideri il seguente frammento di programma che dovrebbe assegnare ad una variabile booleana uguali il valore vero se le due stringhe sono uguali, i valore falso altrimenti

```
. . .
    if lunA == lunB
        for (i=0; i < lunA; i++)
              uguali = a.charAt(i) == b.charAt(i)
    else
        uguali = false;
. . .
```

. . . Esempio . . .

- La soluzione precedente è chiaramente errata perché è sufficiente che due parole della stessa lunghezza abbiano l'ultimo carattere uguale affinché la variabile **uguali** sia posta a vero.
- Pertanto il frammento di programma con le stringhe **'bianco'** e **'giallo'** come dati di ingresso assegnerà alla variabile **uguali** il valore vero
- Si noti che una esecuzione del programma con una scelta casuale dei dati rende molto improbabile la scoperta dell'errore
- Infatti l'errore può esser rilevato solo se i dati di prova soddisfano contemporaneamente le due seguenti condizioni:
  - **lunA** pari a **lunB**
  - **a.charAt(lunA-1)** pari a **b.charAt(lunB-1)**

. . . Esempio

- Con una scelta completamente casuale dei dati la prima condizione è vera con probabilità pari a 1/20 = 0.05 (le stringhe hanno lunghezze comprese fra 1 e 20), mentre la seconda è vera con probabilità pari a circa 1/26 = 0.038 (i caratteri possibili sono 26)
- Pertanto una scelta casuale dei dati di prova ha una probabilità pari a (0.05 x 0.038)=0.0019 di evidenziare l'errore
- Se si eseguono più prove allora la probabilità di trovare l'errore aumenta ma molto lentamente, infatti, la probabilità di trovare l'errore eseguendo cento prove casuali in modo indipendente è inferiore a 0.2

Test a scatola nera

- **Il test a scatola nera di un metodo è basato sulla specifica del metodo**
  - l'insieme dei possibili dati di ingresso per il metodo viene partizionato in sottoinsiemi (insiemi di equivalenza)
  - il metodo viene eseguito utilizzando una sola combinazione dei valori per ciascun insieme di equivalenza
- **L'approccio di test a scatola nera è basato sulla seguente ipotesi**
  - se il metodo si comporta correttamente per una combinazione dei dati di ingresso X scelta nell'insieme di equivalenza C_X, allora il metodo si comporta correttamente anche su ogni altra combinazione dei dati di ingresso scelta nell'insieme C_X

è solo una ipotesi e potrebbe quindi essere sbagliata

Scelta degli insiemi di equivalenza . . .

- **Idea**
  - alcuni insiemi di equivalenza rappresentano dati "**normali**" per il metodo
  - altri insiemi di equivalenza rappresentano dati "**particolari**" per il metodo
    - casi o valori al limite

. . . Scelta degli insiemi di equivalenza

- **Leggi dalla tastiera una sequenza di numeri interi e calcolane la somma**
  - scelta degli **insiemi di equivalenza**
    - la **sequenza è vuota** – caso particolare
    - la **sequenza contiene un solo elemento** – caso particolare
    - la **sequenza contiene più elementi** – caso normale
- **Il test viene poi svolto effettuando una prova per ciascun insieme di equivalenza**
  - scegliendo un insieme di dati rappresentativo per ciascun insieme di equivalenza

Scelta dei dati

- **Leggi dalla tastiera una sequenza di numeri e calcolane la somma**
  - **scelta dei dati per ciascun insieme di equivalenza**
    - **la sequenza è vuota –**
    - **la sequenza contiene un solo elemento – 8**
    - **la sequenza contiene più elementi – 1 2 3 4 5**
  - **che cosa può accadere se la sequenza vale 1 2 3 4 5**
    - **il risultato è 15 – OK**
    - **il risultato è 14 o 10 – errore di uno**
    - **il risultato è 0 – gli elementi non sono stati sommati correttamente**
    - **il programma rimane in attesa di ulteriori dati in ingresso**
  - **sequenze meno significative**
    - **la sequenza vale 0 0 0 0 0**
    - **la sequenza vale 1 1 1 1 1**

Test a scatola trasparente

- **I test a scatola trasparente di un metodo sono basati anche sulla struttura del metodo**
  - **l'idea alla base di questo approccio è che un test può considerarsi concluso solo quando ciascuna istruzione del metodo sia stata eseguita almeno una volta**
  - **questo richiede, ad esempio, di scegliere i dati di ingresso in modo tale che ciascuna istruzione sia eseguita almeno una volta, ed inoltre che ciascuna condizione sia verificata almeno una volta e non verificata almeno una volta**

Metodi di test

- **Un modo comune per effettuare, in pratica, il test di un metodo è definire un metodo di test per il metodo**
  - **sono state scelti gli insiemi di equivalenza per il test e i relativi dati rappresentativi**
  - **un metodo di test per il metodo contiene una invocazione del metodo per ciascuno dei dati scelti**
    - **i parametri attuali dell'invocazione del metodo sono letterali che rappresentano i dati scelti**
    - **il valore restituito dal metodo viene confrontato con quello atteso e/o visualizzato sullo schermo**
    - **un commento motiva la scelta dell'insieme di equivalenza e dei dati di prova**

Documentazione e ripetibilità dei test

- **Se si procede come descritto**
  - il metodo di test costituisce una documentazione della verifica di correttezza effettuata
  - il test è riproducibile e ripetibile
    - se il metodo non passa il test, per la nuova versione del metodo (in cui dovrebbero essere stati individuati e corretti alcuni errori) può essere usato lo stesso metodo di test
    - il test può essere ripetuto quando si definisce una nuova versione del metodo (ad esempio, più efficiente)
- **In generale, non è una buona idea eseguire un test sulla base di dati letti dalla tastiera**
  - il test non è documentato e non è ripetibile
  - ma è l'unica scelta possibile nella verifica di applicazioni interattive

Riferimenti al libro di testo

- **Per lo studio di questi argomenti si fa riferimento al libro di testo, e in particolare al capitolo 16**