

# Il computo metrico con Lua

Roberto Giacomelli

Articolo sul blog <http://robitec.wordpress.com>

e-mail: giaconet dot mailbox at gmail dot com

29 novembre 2011

## Sommario

Certi tipi di opere edili possono essere così articolate da mettere in difficoltà l'utente che intendesse gestirle con un tradizionale software di computazione. In questo post si svilupperà un'idea che può risolvere il problema di efficienza, secondo la quale i dati sono *descritti* da un opportuno linguaggio così da esprimere in modo diretto e concettuale le particolarità dell'opera edilizia.

## Indice

<b>1</b>	<b>Sommario</b>	<b>1</b>
<b>2</b>	<b>Computi metrici</b>	<b>1</b>
<b>3</b>	<b>Il linguaggio dei solai</b>	<b>2</b>
3.1	Osservazioni operative . . . . .	2
<b>4</b>	<b>Elaborare i dati</b>	<b>3</b>
4.1	Elaborazione diretta . . . . .	3
4.2	Elaborazione strutturata . . . . .	4
<b>5</b>	<b>Conclusioni</b>	<b>6</b>
<b>6</b>	<b>Note tecniche</b>	<b>7</b>
<b>7</b>	<b>Licenza ed informazioni varie</b>	<b>7</b>
7.1	Distribuzione/Citazioni . . . . .	7
7.2	Colophon . . . . .	7

## 1 Sommario

Certi tipi di opere edili possono essere così articolate da mettere in difficoltà l'utente che intendesse gestirle con un tradizionale software di computazione. In questo post si svilupperà un'idea che può risolvere il problema di efficienza, secondo la quale i dati sono *descritti* da un opportuno linguaggio così da esprimere in modo diretto e concettuale le particolarità dell'opera edilizia.

## 2 Computi metrici

Un computo metrico di un'opera consiste in una serie di voci che definiscono i singoli componenti edilizi quantificate secondo una specifica unità di misura. Inserire le misure è una di quelle attività leggermente noiose e poco attraenti...

Con un linguaggio specifico è possibile rendere efficiente l'inserimento manuale di queste quantità, in particolare se si desidera che i dati siano raggruppati in livelli strutturati.

### 3 Il linguaggio dei solai

Se l'opera in progetto è composta da diversi edifici a più piani, dovremo calcolare le superfici dei solai raggruppando i dati secondo la doppia classificazione piano/edificio. Saremo così in grado di fornire le superfici di solaio per ciascun piano, quelle totali per singolo edificio e quelle complessive per piano.

Un *linguaggio* semplice che descrive questa struttura informativa potrebbe essere questo:

```
1  solaio{
2    edificio = "A",
3    piano = 1,
4    superficie = 4.50*3.25,
5 }
```

La sintassi che riscontriamo leggendo il frammento di codice, si basa su un costrutto *chiave = valore* ciascuno separato con una virgola e racchiuso in un coppia di parentesi graffe.

I valori testo sono racchiusi tra doppi apici mentre i valori numerici utilizzano il punto e non la virgola per separare la parte decimale. Vi anticipo che il codice precedente corrisponde alla sintassi di [Lua](#), un linguaggio di programmazione libero e multi-piattaforma, più volte trattato sul [blog](#).

Continuando a tagliare su misura il codice per renderlo perfettamente adatto ad esprimere le quantità che vorremmo computare, osserviamo che non ci occorre identificare i singoli solai di uno stesso piano di uno stesso edificio. Alla chiave *superficie* assegniamo allora non un singolo valore o espressione, ma un elenco di espressioni corrispondenti alla geometria dell'impalcato, per esempio (possiamo commentare il codice inserendo due trattini):

```
1  -- cambiamo nome all'oggetto :-)
2  solaiDiPiano{
3    edificio = "A",
4    piano = 1,
5    superficie = {
6      2*4.50*3.25,          -- qui due solai uguali
7      3.80*3.25,
8      4.64*4.20 - 0.40*0.70, -- qui una detrazione di una botola!
9      4.24*(4.20 + 3.50),
10     5.50*3.60,          -- la virgola finale opzionale
11     },                 -- ma rende comode le successive
12 }                      -- aggiunte di valori
```

#### 3.1 Osservazioni operative

L'ambiente in cui si memorizza il codice è un editor di testo ASCII. La comodità sta nel fatto che i dati sono immediatamente disponibili su qualsiasi sistema operativo per eventuali modifiche o verifiche, e possiamo eseguire le operazioni con poca fatica.

Nel frattempo ho aggiunto altri dati come potete vedere dal prossimo frammento di codice:

```
1  solaiDiPiano{edificio="A", piano=1,
2    superficie = {
3      2*(1.60+3.60+2.07)*(4.345+4.245),
4      1.92*2.64,
5      3.60*(0.55+1.05)*2,
6    }
7  }
8
9  solaiDiPiano{edificio="A", piano=2,
10   superficie = {
11     2*(1.60+3.60+2.07)*(4.345+4.245),
12     1.92*2.64,
13     3.60*(0.55+1.05)*2,
14   }
15 }
16
17
18 solaiDiPiano{edificio="A", piano=3,
19   superficie = {
20     2*(1.65+3.60+2.12)*(4.42+4.32),
21     1.92*2.64,
22     3.60*(0.55+1.05)*2,
23   }
```

```

24 }
25
26 solaiDiPiano{edificio="B", piano=1,
27   superficie = {
28     2*(2.54+2.54+2.56+2.545)*(4.345+4.245),
29     2*1.00*2.64
30   }
31 }
32
33 solaiDiPiano{edificio="B", piano=2,
34   superficie = {
35     2*(2.54+2.54+2.56+2.545)*(4.345+4.245),
36     2*1.00*2.64
37   }
38 }
39
40 solaiDiPiano{edificio="C", piano=1,
41   superficie = {
42     2*(1.60+3.60+2.07)*(4.345+4.245),
43     1.92*2.64,
44     3.60*(0.55+1.05)*2,
45   }
46 }
47
48 solaiDiPiano{edificio="C", piano=2,
49   superficie = {
50     2*(1.60+3.60+2.07)*(4.345+4.245),
51     1.92*2.64,
52     3.60*(0.55+1.05)*2,
53   }
54 }
55
56 solaiDiPiano{edificio="C", piano=3,
57   superficie = {
58     2*(1.65+3.60+2.12)*(4.42+4.32),
59     1.92*2.64,
60     3.60*(0.55+1.05)*2,
61   }
62 }

```

## 4 Elaborare i dati

Ma come si elaborano i dati? Il codice scritto corrisponde esattamente ad una serie di chiamate alla funzione Lua `solaiDiPiano()` a cui passiamo una tabella. Eseguendo il codice, verranno calcolate le espressioni delle superfici e le istruzioni contenute nella funzione.

### 4.1 Elaborazione diretta

Proviamo a scrivere la funzione in modo da ottenere la superficie totale dei solai:

```

1  -- superficie totale
2  local totsup = 0
3
4  -- definizione della funzione
5  function solaiDiPiano(s)
6    -- campi in s:
7    -- s.superficie -> array di valori
8
9    -- sommo tutti i valori in s.superficie
10   -- e li inserisco nel totale
11   for _, val in ipairs(s.superficie) do
12     totsup = totsup + val
13   end
14 end
15
16 -- carico il file dati

```

```

D:\>cd test

D:\test>dir
Il volume nell'unità D è DATA
Numero di serie del volume: 284A-7A45

Directory di D:\test

12/12/2011  13.04    <DIR>          .
12/12/2011  13.04    <DIR>          ..
12/12/2011  13.02                452 calctotsup.lua
12/12/2011  13.01                1,119 solai.lua
                2 File                1,571 byte
                2 Directory  144,484,700,160 byte disponibili

D:\test>lua calctotsup.lua
Superficie totale : 1217.299

D:\test>

```

Figura 1: Esecuzione del report per la superficie totale in console

```

17 dofile("solai.lua")
18
19 -- stampo il risultato
20 print(string.format("Superficie totale : %.3f",totsup))

```

Una volta installato sul sistema *Lua*, inseriamo i dati dei solai così come si è deciso di descriverli nel file testo solai.lua. Salviamo in un secondo file il codice che li elabora calcsuptot.lua, ed infine apriamo il terminale o la console ed eseguiamo il comando:

```

1 > lua calcsuptot.lua

```

Fatto! La schermata rappresentativa dell'operazione in Windows è la seguente, dove si può notare l'uso del comando cd per spostarsi nella directory dove avevo salvato i due file.

## 4.2 Elaborazione strutturata

Diamo una definizione strutturata della funzione `solaiDiPiano()` in modo da costruire una tabella dati chiamata **Solai** che potremo leggere in modi diversi. Questa è la via sicuramente più flessibile e potente.

```

1 -- serve un contenitore per i dati:
2 local Solai = {}
3
4 -- creazione della tabella dati
5 function solaiDiPiano(s)
6     -- ricordiamo i nomi dei campi in s
7     -- s.edificio := nome edificio
8     -- s.piano := numero di piano
9     -- s.superficie := array di valori
10
11     -- se non esiste ancora creiamo
12     -- una tabella per edificio
13     -- rispetto al nome
14     -- es.: Solai.A
15     if not Solai[s.edificio] then
16         Solai[s.edificio] = {}
17     end
18
19     -- prendiamoci un riferimento comodo
20     -- alla singola tabella edificio
21     local edificio = Solai[s.edificio]
22
23     -- calcoliamo la somma delle superfici
24     local supiano = 0
25     for _, val in ipairs(s.superficie) do

```

```

C:\ Prompt dei comandi
D:\test>lua report.lua
Edificio A: -----
Piano 1, superficie    141.487
Piano 2, superficie    141.487
Piano 3, superficie    145.416
-----
Tot. sup. edificio A:    428.391
-----
Edificio C: -----
Piano 1, superficie    141.487
Piano 2, superficie    141.487
Piano 3, superficie    145.416
-----
Tot. sup. edificio C:    428.391
-----
Edificio B: -----
Piano 1, superficie    180.258
Piano 2, superficie    180.258
-----
Tot. sup. edificio B:    360.517
-----
Tot. superfici    1217.299
-----
D:\test>

```

Figura 2: Esecuzione del report Piano/Edificio in console

```

26     suppiano = suppiano + val
27     end
28
29     -- intenderemo la tabella edificio
30     -- come un array dove l'indice rappresenta
31     -- il numero di piano ed il valore
32     -- corrispondente, la superficie
33     if edificio[s.piano] then
34         edificio[s.piano] = edificio[s.piano] + suppiano
35     else
36         edificio[s.piano] = suppiano
37     end
38 end

```

Fatto questo, potremo elaborare decine od anche migliaia di elementi si solaio. Tutti i dati saranno memorizzati nella tabella Solaio *Solaio*.

A questo punto scriviamo la funzione di reporting che opererà in lettura. Vogliamo stampare per esempio i totali di piano per edificio:

```

1  -- report piano/edificio
2  function reportPianoEdificio()
3      -- Solai contiene i dati in modo strutturato
4      -- Al primo livello ci sono le tabelle edificio
5
6      local totsup = 0
7      for nomeEd, arraysup in pairs(Solai) do
8          print(string.format("Edificio %s: -----", nomeEd))
9
10         -- stampiamo le superfici piano per piano
11         local totsupEd = 0

```

```

12     for level, sup in pairs(arraysup) do
13         totsupEd = totsupEd + sup
14         print(string.format("Piano %2d, superficie %10.3f", level, sup))
15     end
16
17     totsup = totsup + totsupEd
18     print("-----")
19     print(string.format("Tot. sup. edificio %s: %10.3f", nomeEd, totsupEd))
20     print("-----")
21 end
22 print("-----")
23 print(string.format("Tot. superfici %10.3f", totsup))
24 print("-----")
25 end

```

In console ottengo quello di figura 2:

Se volessimo stampare le superfici piano per piano dovremo costruire una funzione in lettura di questo tipo:

```

1  -- report per piani
2  function reportPiani()
3      -- una tabella per le sup di piano
4      local suppiani = {}
5
6      -- ciclo per singolo edificio
7      -- primo livello della struttura dati Solaio
8      for nomeEd, arraySup in pairs(Solai) do
9          -- ciclo interno di lettura delle sup
10         for piano, sup in pairs(arraySup) do
11             if suppiani[piano] then
12                 suppiani[piano] = suppiani[piano] + sup
13             else
14                 suppiani[piano] = sup
15             end
16         end
17     end
18
19     -- stampa
20     local ts = 0
21     for level, sup in pairs(suppiani) do
22         ts = ts + sup
23         print(string.format("Livello %2d: sup = %10.3f", level, sup))
24     end
25     print("-----")
26     print(string.format("Totale sup = %10.3f", ts))
27 end
28
29 --[[
30 -- l'output con i soliti dati:
31 Livello 1: sup =   463.233
32 Livello 2: sup =   463.233
33 Livello 3: sup =   290.833
34 -----
35 Totale sup =  1217.299
36 --]]

```

## 5 Conclusioni

L'idea proposta è originale? No! Volendo, si possono fare molti esempi tratti dal mondo dell'informatica dove un linguaggio testuale descrive dati strutturati. Con Lua è semplice creare il proprio linguaggio ed elaborare i dati, pensiamo per esempio al caso più complesso riguardante la costruzione dei libretti dei ferri...

Naturalmente, potremo preparare report molto più professionali che non un semplice output in console, per esempio compilando un opportuno sorgente LaTeX per l'uscita in PDF, direttamente con le funzioni Lua di reporting.

Uno svantaggio consiste nella necessità di scrivere qualche linea di codice Lua, ma del resto questo consente la massima libertà di cucire la soluzione sul problema e non è poi così terribilmente difficile. A voi dunque l'onore e l'onere di rendere un po' più attraente la stesura dei computi metrici! Un saluto.

## 6 Note tecniche

Il software occorrente per utilizzare la procedura è il seguente: un editor di testi, consiglio **SciTE** perché in grado di far eseguire il codice premendo semplicemente il tasto F5, e l'interprete **Lua**, quindi tutto software libero, gratuito e legalmente installabile anche in ambito professionale, infine un pizzico di fantasia. Per imparare Lua non vi è niente di meglio del **PiL**. Un saluto.

## 7 Licenza ed informazioni varie

Questo articolo come tutto il materiale didattico/divulgativo del blog <http://robiteX.wordpress.com> è rilasciato sotto licenza Creative Commons “Attribuzione-Non commerciale-Non opere derivate” 2.5 Italia, il cui testo integrale con valore legale è consultabile a [questo indirizzo](#). Ciò significa che:

1. Bisogna sempre attribuire la paternità del materiale a <http://robiteX.wordpress.com>;
2. Non si può usare il materiale per fini commerciali;
3. Non si può alterare o trasformare i contenuti, ne' usarne stralci per creare altre opere.

Se esplicitamente indicato nei commenti iniziali, il codice relativo a programmi software è rilasciato nella specifica licenza.

### 7.1 Distribuzione/Citazioni

Ogni volta che usi o distribuisi parti ridistribuibili quest'opera devi farlo secondo i termini con cui esse sono state rilasciate e avendo cura di comunicare tali termini con chiarezza. Ricorda di inserire sempre un hyperlink alla risorsa che ridistribuisi o citi.

Il modo migliore per dimostrarvi il vostro apprezzamento è semplicemente quello di linkare direttamente le pagine del blog, senza copiare gli articoli in altri siti, oltre naturalmente a lasciare un commento. Ci sono però casi in cui vorreste poter estrapolare alcune parole dai miei articoli per incuriosire i vostri lettori. In quel caso la prassi convenuta è che, grazie a tutti i blogger seri, viene coscienziosamente rispettata, è questa:

- Creare un “blockquote”, ossia un campo in cui è inserita una citazione;
- Inserire nel blockquote solo il primo periodo (le prime poche frasi) di un articolo, diciamo fino ad arrivare al link “Leggi il resto...”;
- Linkare la rimanente parte dell'articolo all'originale su <http://robiteX.wordpress.com>.

Grazie per la collaborazione.

### 7.2 Colophon

Questo documento è stato composto con  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  attraverso uno script in Lua chiamato **wp2pdf** che elabora il file originale *html* del post pubblicato sul blog in WordPress. Si tratta di una versione migliorata del codice pubblicato sul blog stesso. Per saperne di più contattatemi via posta elettronica all'indirizzo nel titolo del documento, o lasciate un commento sul blog.