

Numeri in lettere

Roberto Giacomelli

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

e-mail: giaconet dot mailbox at gmail dot com

21 aprile 2011

Sommario

Nel post viene descritto sia nel linguaggio Java che in Lua, un programma per tradurre un numero nel suo corrispondente letterale tenendo conto delle regole principali della lingua italiana. Si darà poi un'applicazione pratica del codice generato nell'ambito di un documento LuaTeX, il nuovo motore di composizione tipografica erede della tradizione TeX.

Indice

1 Un problema antico: tradurre numeri in lettere	1
2 Soluzione in Java	1
3 Soluzione in Lua	4
4 Soluzione in LuaTeX	5
5 Licenza ed informazioni varie	7
5.1 Distribuzione/Citazioni	7
5.2 Colophon	7

1 Un problema antico: tradurre numeri in lettere

Molti anni fa acquistai un libro che proponeva una serie di esempi, di esercizi, tutti risolti con il linguaggio Basic. Nonostante il fatto che quel libro una volta prestato non venne più restituito (ebbene l'interessato/a ovunque esso/a sia, è ancora in tempo per restituire quel libro e, già che c'è, anche gli altri libri di storia dell'arte, grazie), ricordo che verso la fine esso presentava il problema di tradurre un numero in lettere, per esempio, se l'input fosse stato 123 allora l'output del programma avrebbe dovuto essere "centoventitre".

2 Soluzione in Java

Malauguratamente il linguaggio Java non è in grado di far "ritrovare" i libri "perduti" ai legittimi lettori (anche se credo che Oracle stia lavorando a funzionalità del genere), ma è comunque in grado di dare soddisfazione a chi volesse tradurre vaghi e primordiali ricordi, in una pagina web di un pugno di bit.

Cominciamo allora con qualcosa di molto più sofisticato di quello che si trovava nel codice Basic di quel libro notando che il numero 123 si può tradurre in lettere in tempi successivi scrivendo prima "cento" e continuando con il numero rimanente 23, scrivendo poi "venti" e continuando ancora con il numero 3, traducendolo in "tre". Se pensiamo di unire le tre parole avremo ottenuto "centoventitre".

Se il numero da tradurre in parole fosse invece 6123, dapprima scriveremmo "seimila" e poi ritrovando ancora il numero 123 applicherei la stessa procedura precedente ed alla fine otterrei la sequenza "seimilacentoventitre".

Adotteremo quindi un approccio ricorsivo per cui elaborata la cifra più significativa si lancia la procedura stessa con il numero restante. Il codice che compare di seguito evita i casi di doppia vocale. Applicando strettamente la procedura ricorsiva, il numero 41 diventerebbe "quarantauno" invece di

“quarantuno”. Inoltre, grazie all’overloading dei metodi, il programma è in grado di elaborare numeri decimali traducendoli nel formato letterale previsto per gli importi in euro.

```
1 import java.util.Scanner;
2
3 public class Traduci {
4     // aggiunto il metodo in overloading che fa
5     // corrispondere una cifra in euro alla stringa convenzionale
6     // es.: 58,45 -> "cinquantotto/45"
7
8     public static void main(String[] args) {
9         System.out.print("Digita il numero da tradurre: ");
10        Scanner in = new Scanner(System.in);
11        if (in.hasNextInt()) {
12            System.out.println("in lettere: " +
13                Traduci.NumberToText(in.nextInt()));
14        } else if (in.hasNextDouble()) {
15            System.out.println("in lettere: " +
16                Traduci.NumberToText(in.nextDouble()));
17        } else {
18            System.out.println("Errore nei dati immessi.");
19        }
20    }
21
22    static String NumberToText(int n) {
23        // metodo wrapper
24        if (n == 0) {
25            return "zero";
26        } else {
27            return NumberToTextRicorsiva(n);
28        }
29    }
30
31    static String NumberToText(double importo) {
32        // metodo wrapper
33        int n = (int) Math.round(importo * 100);
34        int parteIntera = n/100;
35        String parteDecimale = String.format("%02d", n%100);
36
37        if (parteIntera == 0) {
38            return "zero/" + parteDecimale;
39        } else {
40            return NumberToTextRicorsiva(parteIntera) + "/" + parteDecimale;
41        }
42    }
43
44    private static String NumberToTextRicorsiva(int n) {
45        if (n < 0) {
46            return "meno " + NumberToTextRicorsiva(-n);
47        } else if (n == 0){
48            return "";
49        } else if (n <= 19){
50            return new String[] { "uno", "due", "tre", "quattro", "cinque",
51                "sei", "sette", "otto", "nove", "dieci",
52                "undici", "dodici", "tredici",
53                "quattordici", "quindici", "sedici",
54                "diciassette", "diciotto", "diciannove" }[n-1];
55        } else if (n <= 99) {
56            String[] vettore =
57                { "venti", "trenta", "quaranta", "cinquanta", "sessanta",
58                "settanta", "ottanta", "novanta" };
59            String letter = vettore[n / 10 - 2];
60            int t = n % 10; // t prima cifra di n
61            // per 1 o 8 va tolta la vocale finale di letter
62            if (t == 1 || t == 8) {
63                letter = letter.substring(0, letter.length() - 1);
64            }
65            return letter + NumberToTextRicorsiva(n % 10);
66        }
67    }
68 }
```

```

66     } else if (n <= 199){
67         return "cento" + NumberToTextRicorsiva(n % 100);
68     } else if (n <= 999){
69         int m = n % 100;
70         m /= 10; // divisione intera per 10 della variabile
71         String letter = "cent";
72         if (m != 8){
73             letter = letter + "o";
74         }
75         return NumberToTextRicorsiva(n / 100) + letter +
76             NumberToTextRicorsiva(n % 100);
77     }
78     else if (n <= 1999){
79         return "mille" + NumberToTextRicorsiva(n % 1000);
80     } else if (n <= 999999){
81         return NumberToTextRicorsiva(n / 1000) + "mila" +
82             NumberToTextRicorsiva(n % 1000);
83     }
84     else if (n <= 1999999){
85         return "unmilione" + NumberToTextRicorsiva(n % 1000000);
86     } else if (n <= 999999999){
87         return NumberToTextRicorsiva(n / 1000000) + "milioni" +
88             NumberToTextRicorsiva(n % 1000000);
89     } else if (n <= 1999999999){
90         return "unmiliardo" + NumberToTextRicorsiva(n % 1000000000);
91     } else {
92         return NumberToTextRicorsiva(n / 1000000000) + "miliardi" +
93             NumberToTextRicorsiva(n % 1000000000);
94     }
95 }
96 }

```

Una volta inserito il codice in un file di testo chiamato "Traduci.java", procedete alla sua compilazione utilizzando uno JDK 5.0 o successivo con il comando (installate eventualmente da voi un JDK scelto tra le varie alternative):

```
1 javac Traduci.java
```

Per lanciare il programma che consiste nel file contenente il bytecode per la macchina virtuale Java ed ha per estensione .class, date il comando:

```
1 java Traduci
```

ed inserite il numero intero o decimale da tradurre in parole. Nell'immagine seguente potete consultare la sessione di compilazione ed esecuzione.

```

roberto@roberto-desktop: ~/Scrivania/numpar
File Modifica Visualizza Terminale Aiuto
roberto@roberto-desktop:~/Scrivania/numpar$ javac Traduci.java
roberto@roberto-desktop:~/Scrivania/numpar$ java Traduci
Digita il numero da tradurre: 488
in lettere: quattrocentottantotto
roberto@roberto-desktop:~/Scrivania/numpar$ java Traduci
Digita il numero da tradurre: 1234,56
in lettere: milleduecentotrentaquattro/56
roberto@roberto-desktop:~/Scrivania/numpar$ java Traduci
Digita il numero da tradurre: 99
in lettere: novantanove
roberto@roberto-desktop:~/Scrivania/numpar$

```

Figura 1: The compile session of the java program

3 Soluzione in Lua

Volgiamo adesso scrivere lo stesso programma in Lua, non tanto per confrontarlo con la versione Java, quanto per predisporre l'uso con Lua_TE_X. Il codice in Lua è molto meno strutturato di quello in Java e ha un sapore, diciamo così, più artigianale ma è più semplice, eccolo:

```
1  #!/usr/bin/lua
2
3  -- script di traduzione di numeri in lettere
4  -- nella lingua italiana
5  -- traduce un numero in lettere
6
7  -- funzione ricorsiva
8  local function NumberToTextRicorsiva(n)
9      if n < 0 then
10         return "meno " .. NumberToTextRicorsiva(-n)
11     elseif n == 0 then
12         return ""
13     elseif n <= 19 then
14         local ventina = { "uno", "due", "tre", "quattro", "cinque",
15                          "sei", "sette", "otto", "nove", "dieci",
16                          "undici", "dodici", "tredici",
17                          "quattordici", "quindici", "sedici",
18                          "diciassette", "diciotto", "diciannove" }
19         return ventina[n]
20
21     elseif n <= 99 then
22         local decine = { "venti", "trenta", "quaranta",
23                          "cinquanta", "sessanta",
24                          "settanta", "ottanta", "novanta" }
25         local letter = decine[math.floor(n/10)-1]
26         local t = n % 10
27         if t == 1 or t == 8 then
28             letter = string.sub(letter,1,-2)
29         end
30         return letter .. NumberToTextRicorsiva(n % 10)
31     elseif n <= 199 then
32         return "cento" .. NumberToTextRicorsiva(n % 100)
33     elseif n <= 999 then
34         local m = n % 100
35         m = math.floor(m/10)
36         local letter = "cent"
37         if m ~= 8 then
38             letter = letter .. "o"
39         end
40         return NumberToTextRicorsiva( math.floor(n / 100)) ..
41             letter ..
42             NumberToTextRicorsiva(n % 100)
43     elseif n <= 1999 then
44         return "mille" .. NumberToTextRicorsiva(n % 1000)
45     elseif n <= 999999 then
46         return NumberToTextRicorsiva(math.floor(n / 1000)) ..
47             "mila" ..
48             NumberToTextRicorsiva(n % 1000)
49     elseif n <= 1999999 then
50         return "unmilione" .. NumberToTextRicorsiva(n % 1000000)
51     elseif n <= 999999999 then
52         return NumberToTextRicorsiva(math.floor(n / 1000000))..
53             "milioni" ..
54             NumberToTextRicorsiva(n % 1000000)
55     elseif n <= 1999999999 then
56         return "unmiliardo" .. NumberToTextRicorsiva(n % 1000000000)
57     else return NumberToTextRicorsiva(math.floor(n / 1000000000)) ..
58             "miliardi" ..
59             NumberToTextRicorsiva(n % 1000000000)
60     end
61 end
```

```

62
63 -- funzione wrapper
64 local function NumberToText(n)
65     if n == 0 then
66         return "zero"
67     else
68         return NumberToTextRicorsiva(n)
69     end
70 end
71
72 -- prelevo il valore numerico come argomento dello script
73 local num = arg[1]
74
75 -- gestisco il caso di numero con decimali
76 local dec = num - math.floor(num)
77 local tail = ""
78
79 if dec ~= 0 then
80     dec = dec * 100
81     tail = "/" .. string.format("%.2d",dec)
82 end
83
84 print("in lettere: ".. NumberToText(math.floor(num))..tail)

```

4 Soluzione in LuaTeX

Adesso vediamo come incorporare il programma in un documento LuaTeX, ovvero in un file di testo che passato come argomento al motore di composizione tipografica LuaTeX, produce un file pdf con il contenuto voluto.

Come per gli altri motori tipografici della famiglia TeX, anche in LuaTeX il testo viene direttamente composto nel documento a meno che non sia preceduto dal carattere di backslash: “\”. Questo particolare carattere viene interpretato come simbolo di *escape* nei confronti del testo che lo segue. Così se TeX incontra il testo:

```
1 Il numero 123456789 in lettere viene scritto come \traduci{123456789}.
```

allora inserirà nel documento finale tutti i caratteri tranne per `\traduci` che viene inteso invece come una *macro*. Questo è il succo di uno dei linguaggi tipografici più noti al mondo.

Dunque basterà associare la corretta definizione della macro `\traduci` per veder comparire automaticamente nel pdf il numero tradotto in lettere scritto tra parentesi graffe con il significato di *argomento*. Il codice Lua deve essere leggermente modificato, primo perché l’operatore modulo corrisponde al carattere di commento di TeX per cui lo esprimeremo in termini matematici (si tratta del resto della divisione intera). Secondo, dobbiamo trasformare i simboli di commento Lua nel simbolo di commento TeX e terzo, occorre premettere al simbolo di tilde che compare nell’operatore di disuguaglianza la macro primitiva `\string` per evitare che venga interpretato diversamente. Ecco il sorgente LuaTeX per intero:

```

1 \directlua{
2 function mod(n,d)
3     return n-d*math.floor(n/d)
4 end
5 }
6
7 \def\traduci#1{%
8 \directlua{
9 local function NumberToTextRicorsiva(n)
10     if n < 0 then
11         return "meno " .. NumberToTextRicorsiva(-n)
12     elseif n == 0 then
13         return ""
14     elseif n <= 19 then
15         local ventina = { "uno", "due", "tre", "quattro", "cinque",
16             "sei", "sette", "otto", "nove", "dieci",
17             "undici", "dodici", "tredici",
18             "quattordici", "quindici", "sedici",
19             "diciassette", "diciotto", "diciannove" }
20         return ventina[n]

```

```

21     elseif n <= 99 then
22         local decine = { "venti", "trenta", "quaranta",
23                         "cinquanta", "sessanta",
24                         "settanta", "ottanta", "novanta" }
25         local letter = decine[math.floor(n/10)-1]
26         local t = mod(n, 10)
27         if t == 1 or t == 8 then
28             letter = string.sub(letter,1,-2)
29         end
30         return letter .. NumberToTextRicorsiva(mod(n,10))
31     elseif n <= 199 then
32         return "cento" .. NumberToTextRicorsiva(mod(n,100))
33     elseif n <= 999 then
34         local m = mod(n,100)
35         m = math.floor(m/10)
36         local letter = "cent"
37         if m \string~= 8 then
38             letter = letter .. "o"
39         end
40         return NumberToTextRicorsiva( math.floor(n / 100)) ..
41             letter ..
42             NumberToTextRicorsiva(mod(n,100))
43     elseif n<= 1999 then
44         return "mille" .. NumberToTextRicorsiva(mod(n,1000))
45     elseif n<= 999999 then
46         return NumberToTextRicorsiva(math.floor(n / 1000)) ..
47             "mila" ..
48             NumberToTextRicorsiva(mod(n,1000))
49     elseif n <= 1999999 then
50         return "unmilione" .. NumberToTextRicorsiva(mod(n,1000000))
51     elseif n <= 999999999 then
52         return NumberToTextRicorsiva(math.floor(n / 1000000))..
53             "milioni" ..
54             NumberToTextRicorsiva(mod(n,1000000))
55     elseif n <= 1999999999 then
56         return "unmiliardo" .. NumberToTextRicorsiva(mod(n,1000000000))
57     else return NumberToTextRicorsiva(math.floor(n / 1000000000)) ..
58         "miliardi" ..
59         NumberToTextRicorsiva(mod(n,1000000000))
60     end
61 end
62
63 % funzione wrapper
64 local function NumberToText(n)
65     if n == 0 then
66         return "zero"
67     else
68         return NumberToTextRicorsiva(n)
69     end
70 end
71
72 % prelevo il valore numerico come argomento dello script
73 local num = #1
74
75 % gestisco il caso di numero con decimali
76 local dec = num - math.floor(num)
77 local tail = ""
78
79 f dec \string~= 0 then
80     dec = dec * 100
81     tail = "/" .. string.format("%.2d",dec)
82 end
83 tex.sprint(NumberToText(math.floor(num))..tail)
84 }}
85
86 Il numero 123456789 in lettere viene scritto come \traduci{123456789}.
87

```

```
88 Ciao
89 \bye
```

Compilatelo dopo averlo inserito in un file chiamato **trad.tex** e compilatelo con il comando da console: **luatex trad** (occorre che una distribuzione recente del sistema T_EX sia installata sul vostro sistema, per esempio T_EX Live).

Se volete potete scaricare da questo [link](#) il pdf risultato.

Come breve riferimento, posso aggiungere che nel codice LuaTeX la definizione della macro `\traduci` si avvale della nuova primitiva `\directlua` che esegue codice Lua, in cui si possono elaborare argomenti rappresentati con il simbolo `# 1` (per il primo di essi), proprio come un normalissimo sorgente T_EX. Inoltre, la funzione di sostituzione dell'operatore modulo viene caricata in memoria in una prima macro `\directlua` dimostrando come in LuaT_EX le varie porzioni di codice distribuite tra le varie `\directlua` del sorgente, condividono lo spazio delle variabili globali.

Ci tengo a precisare che l'operazione di traduzione di numeri in lettere è già stata brillantemente risolta per la lingua italiana dal [Prof. Enrico Gregorio](#) con il suo pacchetto *itnumpar* utilizzando esclusivamente codice T_EX. Questo pacchetto tiene conto degli accenti delle parole delle cifre e della eventuale necessità dell'iniziale maiuscola. Esso è stato scritto per l'elaborazione dei titoli di capitolo per esempio per far uscire il testo "Capitolo Diciotto" anziché del solito "Capitolo 18".

Bene. Ho terminato questo post ricco di codice. Come sempre sono aperti i commenti per aggiungere, segnalare o fare domande. Alla prossima. Ciao.

5 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.

5.1 Distribuzione/Citazioni

Ogni volta che usi o distribuisi parti redistribuibili 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 redistribuisci 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 e 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.

5.2 Colophon

Questo documento è stato composto con L^AT_EX 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.