# Un pacchetto per il controllo del codice fiscale italiano

*Claudio Beccari*

## Sommario

Il codice fiscale, che codifica le generalità della persona a cui si riferisce, deve essere generato dall'amministrazione dello Stato, ma tutti necessitano di un software per controllare di averlo scritto in modo corretto; il software deve quindi essere in grado di verificare che i dati siano coerenti, in particolare deve essere in grado di verificare la correttezza della lettera di controllo finale.

## Abstract

The Italian Fiscal Code is a special string that encodes the personal identity data; it may be issued only by the State Administrative Offices, but when a document is being written a piece of software should be available so as to check the correctness of the encoded string, in particular it must control the final check letter for coherence with the remaining data.

## 1 Introduzione

Il *codice fiscale* è stato introdotto in Italia con il Decreto Ministeriale del 23 dicembre 1976 dal titolo "Sistemi di codificazione dei soggetti da iscrivere all'anagrafe tributaria", (Ministro delle finanze, 1976); esso riguarda non solo le persone fisiche ma anche le persone giuridiche e le altre entità da iscrivere all'anagrafe tributaria anche se prive di personalità giuridica.

Qui mi occuperò del codice usato per le persone fisiche, normalmente conosciuto come *codice fiscale* per antonomasia. Tra l'altro il codice per le persone fisiche viene usato anche come identificativo per i servizi sanitari e svolge una funzione simile ai *social security code* e ai *person mummer* in vigore in molti paesi stranieri

Tutti abbiamo il nostro codice personale e sappiamo che esso è una stringa alfanumerica di 16 lettere e cifre ed ha la forma:

CCCNNNAAAMGGLLLLK

Non tutti abbiamo chiaro come siano legate quelle lettere o cifre ai nostri dati personali; sostanzialmente:

**CCC** sono tre lettere estratte dal cognome;

**NNN** sono tre lettere estratte dal nome;

**AA** sono le ultime due cifre dell'anno di nascita;

**M** è una lettera che identifica il mese di nascita;

**GG** sono due cifre che rappresentano il giorno di nascita, aumentato del valore 40 per le donne, cosicché anche il sesso della persona viene codificato mediante questa aggiunta;

**LLLL** sono una lettera seguita da tre cifre che codificano il luogo di nascita: il codice del comune per i nati in Italia o il codice della nazione per i nati all'estero;

**K** è una lettera di controllo che viene determinata in base ai 15 caratteri precedenti; è una specie di "prova del nove" per controllare la correttezza della stringa che forma l'intero codice fiscale.

Per evitare *omocodie*, cioè i rari casi in cui due o più persone hanno i primi quindici caratteri uguali (e quindi anche il sedicesimo carattere di controllo), le sette cifre che compaiono nel codice possono venire sostituite con lettere, secondo regole specificate nel Decreto citato ma, evidentemente, solo l'amministrazione centrale dello Stato può sapere se un dato codice sia già stato assegnato a chi e quando, in modo da poter eseguire le sostituzioni senza generare duplicati.

Spesso capita di scrivere documenti nei quali è necessario riportare il codice fiscale; non sempre si ha a disposizione il tesserino dell'Agenzia delle Entrate o la Tessera Sanitaria; ma anche disponendone capita di commettere degli errori di digitazione.

Il pacchettino che presento qui, da usare con il markup di LaTeX, serve appunto per verificare che la lettera di controllo K sia conforme con gli altri dati secondo l'algoritmo specificato nel Decreto citato. Questo non assicura che il codice sia corretto in ogni suo dettaglio, perché sono sempre possibili errori che si compensano, ma l'eventualità che questa verifica dia un falso positivo è molto remota.

## 2 L'algoritmo di verifica

Ad ogni carattere, lettera o cifra, che compare nelle prime 15 posizioni della stringa viene assegnato un valore numerico a seconda della sua posizione nella stringa; si esegue poi la somma di questi valori e se ne calcola il valore modulo 26; cioè si calcola il resto della divisione intera fra questa somma e il numero 26; il valore modulare è un numero compreso fra 0

Tabella 1: Valori da assegnare ai primi 15 caratteri della stringa del codice fiscale

| Carattere | Posizione pari | Posizione dispari |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |
| 2 | 2 | 5 |
| 3 | 3 | 7 |
| 4 | 4 | 9 |
| 5 | 5 | 13 |
| 6 | 6 | 15 |
| 7 | 7 | 17 |
| 8 | 8 | 19 |
| 9 | 9 | 21 |
| A | 0 | 1 |
| B | 1 | 0 |
| C | 2 | 5 |
| D | 3 | 7 |
| E | 4 | 9 |
| F | 5 | 13 |
| G | 6 | 15 |
| H | 7 | 17 |
| I | 8 | 19 |
| J | 9 | 21 |
| K | 10 | 2 |
| L | 11 | 4 |
| M | 12 | 18 |
| N | 13 | 20 |
| O | 14 | 11 |
| P | 15 | 3 |
| Q | 16 | 6 |
| R | 17 | 8 |
| S | 18 | 12 |
| T | 19 | 14 |
| U | 20 | 16 |
| V | 21 | 10 |
| W | 22 | 22 |
| X | 23 | 25 |
| Y | 24 | 24 |
| Z | 25 | 23 |

e 25, e serve per identificare la lettera dell'alfabeto "latino" di 26 lettere che occupa quella posizione (A corrisponde a 0, B a 1, ..., Z a 25); questa è la lettera di controllo.

La parte complicata consiste proprio nell'assegnazione del valore ad ogni carattere a seconda della sua posizione: bisogna rifarsi alla tabella 1.

Si noti che la lettera di controllo, occupando la sedicesima posizione, è in posizione pari, quindi anch'essa corrisponde alle indicazioni della tabella 1, solo che nel suo caso se ne conosce il valore e bisogna determinare la lettera.

L'algoritmo non è complicato, ma fare i conti a mano è sempre fonte di possibili errori anche se si eseguono le somme con l'aiuto di una calcolatrice; un programmino *ad hoc* risulta quindi sempre utile.

## 3 Il pacchetto codicefiscaleitaliano

Il seguente pacchetto realizza l'algoritmo di controllo del codice fiscale. Si noti che il motore di composizione, sia esso pdflatex o xelatex, non è particolarmente adatto per eseguire calcoli; probabilmente context mkiv e lualatex sono attrezzati meglio, ma la stragrande maggioranza degli utenti usa quasi esclusivamente il primo programma pdflatex e alcuni usano xelatex; gli altri due programmi citati sono usati prevalentemente da professionisti e da persone in grado di sfruttare appieno il linguaggio di scripting Lua, ma non rappresentano ancora la maggioranza degli utenti del sistema TEX.

I problemi che l'algoritmo presenta sono quelli di determinare il valore da associare ad una lettera a seconda della sua posizione e il passaggio inverso dal risultato dei calcoli alla lettera di controllo; le somme e le altre operazioni fra numeri interi non sono un problema per il motore di composizione che, appunto, è costruito per lavorare *solo* con l'aritmetica intera, tanto che, in altre applicazioni, per eseguire calcoli con i numeri fratti bisogna ricorrere a "sporchi trucchi", che D.E. Knuth stesso chiama *dirty tricks*. Qui almeno non abbiamo bisogno di sporchi trucchi per eseguire i calcoli, ma ne necessitiamo per eseguire le conversioni dai caratteri ai numeri e viceversa.

Per fare questo sono dovuto scendere a livello di comandi primitivi del sistema TEX; poi ho usato dei costrutti estratti dal nucleo di LATEX; mi sono servito sia delle espressioni numeriche primitive sia di quelle introdotte con l'estensione ε-TEX che, ormai da diversi anni, è integrata con tutti i motori di composizione; userò anche i registri numerici, i contatori, identificati con numeri superiori a 255, e mi servirò dei comandi primitivi per assegnare dei nomi a questi contatori, ma non ne allocherò nessuno né con `\newcount` né con `\newcounter`, mantenendo tutto l'algoritmo dentro ad un gruppo, cosicché alla fine ogni contatore e ogni variabile usata riprende il suo valore precedente.

Il codice è il seguente, ma lo commenterò facendo riferimento ai numeri di riga, non necessariamente nell'ordine in cui il codice è scritto.

```
\NeedsTeXFormat{LaTeX2e}[2011/06/20]
\ProvidesPackage{codicefiscaleitaliano}%
[2012/05/06 v.1.2 Controlla il codice
    fiscale italiano]
\newif\ifcontrollo \controllotrue
\newif\ifstampacodice \stampacodicefalse
\def\getCFletter#1#2!{%
\ifx#1\space\getCFLetter#2!\else
\Letter=‘#1\def\CFisc{#2}\fi}
\def\getOddValore{%
\ifnum\Letter<\A
\valore=\expandafter
  \ifcase\numexpr\Letter-\zero\relax
```

```
1\or0\or5\or7\or9\or13\or15\or17\or19
  \or21\fi
\else
\valore=\expandafter
  \ifcase\numexpr\Letter-\A\relax
1\or0\or5\or7\or9\or13\or15\or17\or19%
\or21\or2\or4\or18\or20\or11\or3\or6\or8%
\or12\or14\or16\or10\or22\or25\or24\or23
\fi\fi}
\def\getEvenValore{%
\ifnum\Letter<\A
\valore=\numexpr\Letter-\zero\relax
\else
\valore=\numexpr\Letter-\A\relax
\fi}
\newcommand*\codicefiscaleitaliano{%
\@ifstar{\c@dfiscit[\stampacodicetrue]}
  {\c@dfiscit}}
\newcommand*\c@dfiscit[2]
  [\stampacodicefalse]{#1\edef\CFisc{#2}%
\let\codfisc\CFisc
\begingroup
\countdef\cifra=256 \cifra=\z@
\countdef\A=258\A=`\A
\countdef\zero=260 \zero=`\0
\countdef\Letter=262
\countdef\valore=264
\countdef\somma=266 \somma=\z@
\@whilenum\cifra<16\do{\advance\cifra\@ne
\ifx\CFisc\@empty
\cifra=16\controllofalse
\PackageError{codicefiscaleitaliano}{%
\MessageBreak
**************************************
\MessageBreak
Il codice fiscale #2\MessageBreak
non ha 16 caratteri.
\MessageBreak
L'esecuzione della verifica viene
\MessageBreak interrotta.\MessageBreak
**************************************
}{Premere <invio> per continuare}%
\else
\expandafter\getCFletter\CFisc!\relax
\ifodd\cifra
    \getOddValore%
\else
    \getEvenValore%
\fi
\advance\somma\valore\fi}%
\unless\ifx\CFisc\@empty\controllofalse
\PackageError{codicefiscaleitaliano}{%
\MessageBreak
**************************************
\MessageBreak
Il codice immesso #2\MessageBreak
contiene piu' di 16 caratteri.
\MessageBreak
L'esecuzione della verifica viene
\MessageBreak interrotta.\MessageBreak
**************************************
}{Premere <invio> per continuare.}%
\else
\ifcontrollo
\advance\somma-\valore
\Letter\somma
\divide\Letter by 26\relax
\somma=\numexpr\somma - 26*\Letter\relax
\ifnum\valore=\somma
\PackageInfo{codicefiscaleitaliano}{%
\MessageBreak Codice fiscale OK}
\else
\controllofalse
\PackageError{codicefiscaleitaliano}{%
\MessageBreak
**************************************
\MessageBreak
Codice fiscale #2 errato\MessageBreak
**************************************
}{Premi S oppure Q oppure <invio>;
\MessageBreak
il file verra' elaborato lo stesso ma
\MessageBreak
il codice fiscale deve venire
ricontrollato!\MessageBreak
**************************************}
\fi\fi\fi
\ifcontrollo\ifstampacodice\codfisc\fi\fi
\endgroup}
\endinput
```

Il cuore dell'algoritmo si trova nelle righe 57–63: con l'aiuto del contatore `\cifra`, che punta alla posizione specifica del carattere nella stringa del codice fiscale, si esegue un ciclo ripetitivo mediante `\@whilenum` per esaminare uno alla volta i caratteri del codice; mediante `\getCFletter` si estrae il valore numerico ASCII del carattere estraendolo dalla stringa del codice; questa stringa del codice si accorcia via via e si riduce a una stringa vuota al sedicesimo carattere estratto; questo, però, non influenza il valore del codice che l'utente ha dato in pasto all'algoritmo, se non altro perché il tutto viene eseguito dentro il gruppo aperto con `\begingroup` nella riga 35 e che viene chiuso con `\endgroup` nella riga 102.

In base alla parità del numero contenuto nel contatore `\cifra` vengono eseguite le macro `\getOddValore` oppure `\getEvenValore` che determinano il valore da assegnare al carattere in base alla tabella 1; questo valore viene caricato nel contatore `\valore` e subito dopo si esegue la somma nel contatore `\somma`; si noti che la somma contiene anche l'addendo corrispondente alla lettera di controllo, la sedicesima esaminata. Il valore di questa sedicesima lettera è contenuto nel contatore `\valore` all'ultimo giro, quindi lo si può sottrarre (e lo si farà dopo alcuni controlli nella riga 78) e lo si può usare per controllare se esso ha il valore che

dovrebbe avere se il codice fiscale è giusto, cioè se il suo valore corrisponde alla somma modulo 26 dei primi quindici caratteri.

In effetti nella riga 79 si usa il contatore `\Letter`, non più necessario, ma utile per conservare il quoziente della divisione per 26 troncato per difetto all'intero più vicino; nelle righe 80–81 questo quoziente viene moltiplicato per 26 e il suo risultato viene sottratto dal contatore `\somma` per determinare il resto, cioè il valore modulare, raccolto ancora nel contatore `\somma` il cui precedente valore complessivo non interessa più.

Ecco: a questo punto in `\valore` abbiamo il valore della lettera di controllo associata al codice da controllare e in `\somma` abbiamo il valore modulare che ci consentirebbe di determinare la lettera di controllo giusta; se i valori numerici contenuti in `\valore` e `\somma` sono uguali, i dati immessi per il controllo danno esito positivo, altrimenti viene emesso un messaggio di errore; il codice introdotto non viene modificato, ma l'utente viene avvisato che il codice che ha introdotto è errato e sta a lui scoprire dove ha sbagliato a scrivere il codice; sottolineo che non sarebbe corretto cambiare la lettera di controllo del codice introdotto solo per renderlo coerente con l'algoritmo di verifica; l'errore potrebbe essere altrove e quindi fare una cosa del genere vorrebbe dire introdurre un secondo errore per compensare gli effetti del primo.

Si noti che il comando riservato all'utente può essere asteriscato, come si vede nelle righe 29–31. Il comando senza asterisco semplicemente esegue il controllo; se non ci sono errori, nessuno comando permette di scrivere il codice giusto nel documento che si sta allestendo; se invece si usa il comando asteriscato e le verifiche sono positive, il codice dato in pasto al comando viene trascritto nel documento.

Un altro dettaglio: invece di dare la stringa di sedici lettere tutte di seguito è consentito dare la stringa intercalata da spazi, per renderne più esplicite le sue varie parti ma anche per scriverne il codice completo nel documento che si sta allestendo consentendo, grazie agli spazi, di evitare righe mal composte con una parte del codice fiscale che sporge nel margine (Overfull hbox); questi spazi sono irrilevanti per l'algoritmo di verifica, che semplicemente li salta; si veda più avanti l'esempio d'uso.

Le altre macro definite nelle righe che non ho ancora commentato sono ancillari rispetto all'algoritmo principale. In particolare nelle righe da 36 a 41 si definiscono dei nomi per i contatori numerati con i numeri pari da 256 a 266, superiori a 255, (cosa che sarebbe impossibile fare senza le estensioni di ε-TEX); in realtà non è così importante usare questi contatori aggiuntivi, perché comunque tutto rimane locale dentro il gruppo, ma è una abitudine che io ho e che mi dà la sicurezza di non interferire nemmeno per sbaglio con i contatori usati normalmente dalle macro del nucleo di LaTeX.

Nelle righe 7–8 viene definita la macro ad argomenti delimitati`\getCFletter`: essa riceve la stringa del codice fiscale eventualmente accorciata da prelievi precedenti; la stringa viene terminata con un punto esclamativo nella chiamata; il primo token della stringa, cioè il primo carattere, viene prelevato come argomento `#1`, mentre tutti gli altri vengono assegnati all'argomento `#2` che poi la macro usa per ridefinire la nuova stringa accorciata dal prelievo del primo carattere; si noti che se questo primo carattere è uno spazio la macro richiama se stessa, in modo da non tenere conto dello spazio; questo consente di intercalare degli spazi nella stringa alfanumerica di 16 caratteri del codice in modo da separarne visivamente le sue componenti.

Ma l'assegnazione `\Letter=‘#1` costituisce lo "sporco trucco" con il quale il codice ASCII del carattere che costituisce il primo argomento viene assegnato al contatore `\Letter`. Normalmente, e lo si è fatto nelle assegnazioni delle righe 37 e 38, non si usa il backtick da solo ma si usa la sequenza `‘\` per estrarre il codice ASCII di un carattere. Se si fosse scritto `Letter=‘\#1`, il contatore `\Letter` sarebbe stato caricato con il codice ASCII del segno `#`, non dell'argomento `#1`.

Nelle righe da 23 a 28 viene definita la macro per assegnare il valore ai caratteri di posto pari: per le cifre basta prendere il codice della cifra e sottrargli il codice dello zero; per le lettere basta prendere il codice della lettera e sottrargli il codice della lettera 'A'; cosicché le assegnazioni del valore ai caratteri di posto pari è decisamente semplice, come per altro è evidente anche dalla seconda colonna della tabella 1.

Invece dalla riga 10 alla riga 22 si trova la definizione della macro `\getOddValore`, molto più complessa, perché in base al codice ASCII della lettera da esaminare a cui bisogna sottrarre il codice di zero o di 'A' a seconda che sia una cifra o una lettera, si può generare un puntatore per far lavorare il costrutto `\ifcase...\or...\or...\fi` al fine di estrarre il valore giusto dalla tabella corrispondente alla terza colonna della tabella 1.

Diversi test vengono eseguiti e oltre al messaggio relativo alla inesattezza del codice, altri due testi informano se il codice immesso è più corto o più lungo di 16 caratteri.

## 4 Commenti

Il pacchettino è stato controllato nel senso che ho controllato diversi codici fiscali; per esempio, dopo aver caricato il pacchetto codicefiscaleitaliano.sty nel preambolo del documento, per controllare il codice fiscale di Maria Angelini, nata ad Urbino il 24 agosto 1908, NGLMRA08M64L500N, basta dare il comando :

```
\codicefiscaleitaliano{NGLMRA08M64L500N}
```

anche nella forma:

```
\codicefiscaleitaliano
                    {NGL MRA 08M64 L500 N}
```

e viene solo trascritto nel file `.log` che quel codice fiscale è OK; se invece si fosse commesso un qualunque errore di battitura, l'esecuzione del programma di composizione si sarebbe arrestato con un vistoso messaggio d'errore che non può sfuggire all'utente più distratto, in modo che possa provvedere alle correzioni o alle verifiche necessarie. Si noti ancora che se si usa il comando asteriscato il pacchetto, oltre alla verifica, scrive anche il codice nella stessa forma come lo si è immesso nell'argomento del comando; per esempio si ottiene: NGL MRA 08M64 L500 N se si usa il codice scritto come indicato poco sopra.

Bisogna osservare anche che tutta la sintassi dei comandi primitivi è descritta nel TEXbook, (Knuth, 1984); siccome però si sono usati dei comandi "più primitivi" del solito, è possibile che anche un utente esperto non ci si ritrovi se non rivedendo con pazienza tutte le informazioni necessarie sul TEXbook.

Ho scritto questo articoletto solo per mostrare un'altra cosa insolita fra le tante che si possono fare con il sistema TEX; penso e spero che non si tratti solo di un mio passatempo, ma che la cosa possa essere effettivamente utile, specialmente per quegli utenti italiani del sistema TEX che scrivono documenti legali, fiscali e simili, dove è necessario scrivere spesso i codici fiscali.

## Riferimenti bibliografici

Knuth, D. E. (1984). *The TEXbook*. Addison Wesley, Reading, Massachusetts, 18ª edizione.

Ministro delle finanze (1976). «Sistemi di codificazione dei soggetti da iscrivere all'anagrafe tributaria». Supplemento ordinario della Gazzetta Ufficiale n. 345 del 29 dicembre 1976.

▷ Claudio Beccari
Villarbasse (TO)
`claudio dot beccari at gmail dot com`