The extension package codicefiscaleitaliano*

Claudio Beccari

Contents

Abstract

This small package is meant to help users to check the Italian Fiscal Code spelling; it produces an "info" message in the log file if the code is selfconsitent; otherwise, without actually correcting the code, it issues an error message that informs the user of the incorrecteness of the current fiscal code and urges him to provide for a revision, but it does not forbid to continue typesetting the document.

This package should be helpful for everybody who typesets legal or financial documents for the Italian Administration Offices; therefore the user either knows Italian or his mother language is Italian; therefore most of the user guide part will be typeset in Italian.

1 Introduzione

Chiunque abbia bisogno di scrivere il codice fiscale di una persona fisica trova grande giovamento se può disporre di un software che gli controlli l'ortografia del codice; basta il più banale refuso che il codice perde ogni validità.

Questo pacchetto verifica solo la lettera di controllo, che dipende, secondo un apposito algoritmo, dai precedenti 15 caratteri che formano il codice: è una specie di prova del nove che permette non di trovare l'errore, ma di verificare l'inconsistenza dei caratteri che formano la stringa.

Come è noto, il codice fiscale delle perone fisiche è formato da sedici caratteri secondo lo schema seguente:

CCCNNNAAMGGLLLLK

dove:

CCC sono tre lettere estratte dal cognome della persona;

*Version number v.1.2; last revised 2012/05/06.

NNN sono tre lettere estratte dal nome della persona;

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

M è una lettera che identifica il mese di nascita della persona;

GG sono le cifre del giorno di nascita della persona, eventualmente aggiunte al numero 40, nel caso delle persone di sesso femminile;

LLLL è il codice indicativo del comune o della nazione straniera di nascita della persona, a seconda che sia nata in Italia o all'estero;

K è la lettera di controllo.

Per evitare situazioni di *omocodia*, cioè di persone identificate dallo stesso codice, le sette cifre che compaiono normalmente nel codice possono venire sostituite da apposite lettere, ma solo l'Amministrazione Centrale dello Stato può farlo perché solo l'amministrazione conosce quali codici siano già stati attribuiti, a chi e quando. All'utente finale non resta altra possibilità che verificare mediante la lettera di controllo la coerenza dell'intero codice, ma non di generare codici solo a partire dalle generalità di una persona; o meglio, non sarebbe molto complicato farlo, ma si potrebbe solo generare il codice di default, non un codice che garantisca di evitare l'omocodia.

2 Uso del pacchetto

Il pacchetto si carica nel solito modo, mediante `\usepackage`; non ci sono opzioni da esprimere. Lo si usa semplicemente inserendo nel testo da comporre il comando

`\codicefiscaleitaliano{`⟨*codice fiscale da controllare*⟩`}`

Il ⟨*codice fiscale da controllare*⟩ va scritto in lettere maiuscole. Come impostazione predefinita il comando `\codicefiscaleitaliano` esegue solo la verifica della coerenza del codice immesso. La versione asteriscata `\codicefiscaleitaliano*` produce anche la stampa del codice verificato.

Eseguito il controllo il pacchetto si limita di scrivere un messaggio nel file log per ricordare quale codice si è controllato, se l'esito del controllo è positivo. Se il codice fosse inconsistente, invece, il pacchetto scrive un vistoso messaggio di errore indicando il codice inconsistente; non impedisce di continuare a comporre il documento, ma inviita l'utente a ricordarsi di verificare e correggere l'errore. Ovviamente il pacchetto non "corregge" la lettera di controllo, perché l'errore potrebbe essere in uno o più degli altri 15 caratteri, per cui correggere la lettera di controllo potrebbe consistere nell'aggiungere errore ad errore, producendo un codice con almeno due errori, del tutto invalido ai fini legali; probabilmente potrebbe dare luogo a qualche altro reato, a seconda del documento che si sta scrivendo.

Esempio: `\codicefiscaleitaliano{NGLMRA08M64L500N}`

Siccome il codice è valido, non appaiono messaggi sullo schermo; se fosse stato invalido, sarebbe apparso un vistoso messaggio d'errore sullo schermo.

Finché non si controlla un altro codice, il codice corrente è ancora disponibile nella macro `\CFisc` che può essere usata per scrivere il codice senza doverlo ribattere. Infatti ottengo: NGLMRA08M64L500N.

Peraltro il comando `\codicefiscaleitaliano*{NGLMRA08M64L500N}` produce direttamente NGLMRA08M64L500N. Ma il comando asteriscato non produce niente se il codice fiscale non è consistente, mentre entrambi i comandi definiscono comunque la macro `\CFisc`, il cui uso produce il codice fiscale immesso per essere verificato sia esso consistente oppure errato.

Attenzione: devono essere prese alcune precauzioni:

1. Il codice fiscale da controllare deve essere immesso *solo* in lettere maiuscole, però può avere i suoi campi spaziati e gli spazi non contano nella determinazione della lunghezza della stringa da controllare. Infatti il codice `\codicefiscaleitaliano*{NGL MRA 08M64 L500 N}` produce nel file log l'informazione che il codice fiscale è consistente e nel testo ripete la stringa spaziata immessa per il controllo: NGL MRA 08M64 L500 N.

2. Se il codice immesso ha meno di 16 caratteri, il comando produce un avviso di errore e non esegue nessun controllo.

3. Se il codice immesso ha più di 16 caratteri il comando produce un avviso di errore e non esegue nessun controllo.

3 Disclaimer

Although I did my best so as to assure a correct control of the fiscal code string, I cannot be absolutely sure that this control does not produce false positive or false negative results.

Since this work is subject to the LaTeX Project Public Licence, it is the case to redirect the user to the full text of this license; here I reproduce only this section:

4 No Warranty

There is no warranty for the Work. Except when otherwise stated in writing, the Copyright Holder provides the Work 'as is', without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the Work is with you. Should the Work prove defective, you assume the cost of all necessary servicing, repair, or correction.

In no event unless agreed to in writing will the Copyright Holder, or any author named in the components of the Work, or any other party who may distribute and/or modify the Work as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of any use of the Work or out of inability to use the Work (including, but not limited to, loss of data, data being rendered inaccurate, or losses sustained by anyone as

a result of any failure of the Work to operate with any other programs), even if the Copyright Holder or said author or said other party has been advised of the possibility of such damages.

5 The documented code

Here begins the code documentation; I warn the reader that this code contains some "dirty tricks", at least according to my opinion; therefore the reading might result quite difficult. I try to be clear also for my own behalf, because I would like to be able to understand it when I read it some time from now.

In the preamble of this documented TEX file we requested a recent LATEX kernel; actually this code contains very little LATEX code, but it requires a sufficiently recent typesetting engine, therefore a sufficiently recent LATEX kernel; most of the code contained hereafter is primitive TEX code.

The control algorithm requires that each fiscal code string character be examined and, according to its position in the code (even or odd position order), be assigned a specific value according to a table established by the law: see table 1.

Once assigned the above numeric value to each one of the first fifteen characters of the code, it is necessary to sum them up and compute this sum's modulo 26 value; this value lies in the interval from zero to 25, and it is used to look up in the second column of table 1 which letter corresponds to such a number; this is the computed control letter, and the sixteenth character of the fiscal code string subject to testing must match such computed control letter.

We have to produce macros to detach from left to right one character at a time from the fiscal code string; we have to determine its even or odd position, assign it a value and sum up all such values. Such macros should verify also if the input fiscal code is long exactly 16 characters long. In particular I prefer to do the following: The recursion goes on for 16 cycles; if before the end of the full recursion the remaining string is void, an error message is issued and code is produced to exit the recursion and the whole procedure. Similarly, if at the end of the recursion the string is not void, another error message is issued and the procedure is exited. We need a couple of logical switches to perform the whole procedure.

```
\newif\ifcontrollo \controllotrue
\newif\ifstampacodice \stampacodicefalse
```

First we define a delimited argument macro that detaches the leftmost character and retrieves its ASCII address, while reassigning the remaining characters to the possibly shortened fiscal code string; at the end of the process the fiscal code string should vanish.

Notice that with the input string right delimited by an exclamation mark, only the first token (the first character) is assigned to argument `#1`, and the remaining string to the second argument `#2`. Notice also that `\Letter` is a TEX counter and the notation `‘#1` retrieves the ASCCI address corresponding to argument `#1`. It would be wrong to write `‘\#1` because the `\Letter` counter would be assigned the ASCII address of `#`.

Table 1: Values to be assigned to the single characters in the fiscal code string according to their position

Character	Even posisiton	Odd position
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

```
\def\getCFletter#1#2!{\ifx#1\space\getCFLetter#2!\else
\Letter=‘#1\def\CFisc{#2}\fi}
```

Now we generate a macro that assigns counter`\valore` a specific value for odd positioned characters, according to the third column of table 1. We first check if the ASCII address of the character precedes that of character ‘A’; in this is true, we are dealing with a digit, otherwise with a letter. Since the third column of table 1 does not contain ordered values we need a look up table: even better, we need a numeric expression that yields a value between zero and 25 depending on the ASCII address of the examined character: such an address is contained in counter `\Letter` therefore we have to subtract the ASCII address of zero or of ‘A’ depending if the character is a digit or a letter; this is the purpose of the `\numexpr` numerical expressions performed with the ε-TEX syntax:

```
\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}
```

The corresponding macro for even positioned characters is much simpler, because the values to be assigned the counter `\valore` are simply ordered.

```
\def\getEvenValore{%
\ifnum\Letter<\A
\valore=\numexpr\Letter-\zero\relax
\else
\valore=\numexpr\Letter-\A\relax
\fi}
```

Now comes the real testing macro. First the fiscal code string to be checked is assigned to the control sequence `\CFisc`; then a group is opened so that any assignment performed within this group remains local and everything is undone upon closing the group. Several counters are assigned a name; the choice of using counter numbers above 255 is a habit of mine since the time when I first discovered this ε-TEX feature is permanently included within the modern distributions of the TEX system; apparently this dates back to the year 2005, so that it may be assumed that now every TEX distribution in use complies with this upgrade. In any case since we used numerical expression by means of `\numexpr`, this upgrade must be operating, otherwise we'd had already some errors. Of course this package might control by itself if the ε-TEX features are active, but we hope they are since they have been available for so may years.

```
\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@
```

Then we start a “while”...‘do’...” cycle by resorting to the LaTeX kernel macro `\@wiledo`; we use the counter `\cifra` to count and point to the position of a character and we cycle trough all the 16 fiscal code characters; on the first run when `\cifra` equals zero, the immediate stepping up by 1 assures that the pointer has always the correct value and parity; the last cycle is entered with `\cifra` holding the value 15, so the cycle is executed, but it is immediately stepped up to 16; this implies that even the sixteenth character is extracted from the fiscal code string (leaving it void) and its even position value is computed in the same way as all other even positioned characters. But this value is added to the total sum within the cycle. Since this should not be done, upon exiting the cycle, we must subtract the last character value from the sum, in order to execute the modular arithmetics on the sum of the first fifteen character values, while at the same time retaining the value associated to the sixteenth character of the fiscal code string to be checked.

```
\@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
```

Now we proceed with the actual verification. The input string control letter code is stored in counter `\valore`; we now determine the modular value of the computed sum over the values of the first 15 characters; we use the counter `\Letter`, that we don't need anymore, as a buffer for the integer quotient of the division by 26; we have to use the primitive TEX integer division because the one executed by the ε-TEX extensions *rounds* the result to the nearest integer, instead of *truncating* it to the nearest lower integer. This done we calculate the integer remainder of the division with a `\numexpr` clause; the result is the value the control letter should have to be consistent with the first 15 characters of the fiscal code string; if these two values do agree, then the fiscal code string is consistent otherwise it is a faulty string, may be with a typo, or just incorrectly read from the plastic card issued by the Ministry of Finance Agency of Incomes, or by any other document. Suitable messages are therefore issued.

```
\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>; il file
verra' elaborato lo stesso ma il codice
fiscale deve venire ricontrollato!}
\fi
\fi
\fi\ifcontrollo\ifstampacodice\codfisc\fi\fi\endgroup}

\endinput
```

Please, be careful if the fiscal code string is found consistent: it might contain two typos whose effects compensate each other. In any case do not blame me: I did my best to implement the checking algorithm according to the fiscal code legislation; do not blame TEX and company either, because they do their computations the best they can. If it is possible that two or more errors compensate their effects, this is due to the way the fiscal code has been defined.