Una tabella che fa calcoli

Roberto Giacomelli

Sommario

Una semplice sintassi di markup del testo è uno dei punti di forza di LaTeX. L'utente è in grado di costruire il contenuto informativo del documento senza doversi preoccupare di come esso stesso apparirà sulla pagina.

L'articolo vuole mettere alla prova questa caratteristica importante per la produttività dell'autore e per la qualità del suo lavoro costruendo, con particolare attenzione al progetto della sintassi, un nuovo ambiente di tipo tabella utile per fatture, note di spesa o di liquidazione.

Si discuteranno prima gli aspetti linguistici, dando conto di una sperimentazione sul campo condotta coivolgendo un gruppo di utenti in àmbito aziendale, e poi il codice completo del nuovo ambiente, chiamato calctab, iniziando dal problema del calcolo in virgola mobile in TeX e dall'elenco dei potenti pacchetti LaTeX impiegati.

Si daranno infine alcuni spunti metodologici per l'incremento della diffusione di LaTeX nelle aziende e negli studi professionali.

Abstract

One of main advantages of LaTeX is an easy markup syntax of the text. The user is able to build up the informative content of the document without worrying about how it will appear on the page.

This paper aims at testing this remarkable characteristic of LaTeX, both for the author's productivity and the quality of his work, bulding, with particular attention to the syntax project, a new environment of type table for invoices, expense notes and liquidation.

The linguistic aspects will be first discussed, taking into account a testing carried out on a group of users working in the business sector. Then the complete code of the new environment called calctab, will be proposed, starting with the list of the powerful LaTeX packages, it is based upon, and the numeric TeX capability.

In the end some method hints will be given in order to widespread LaTeX in business and professional offices.

1 Linguaggio: sintassi e semantica

La sintassi base di LaTeX è particolarmente semplice, basandosi essenzialmente sulla seguente struttura: un carattere di controllo che precede il nome del comando, cui segue il testo dell'argomento racchiuso tra parentesi graffe[1].

Durante il processo di compilazione verrà eseguito il codice corrispondente al nome del comando utilizzando come input il testo dell'argomento e ciò produrrà il risultato sulla pagina.

L'utente deve quindi conoscere non solo i nomi dei comandi corrispondenti alle procedure di tipo composizione che desidera effettuare, ma anche il significato che verrà attribuito al testo dell'argomento e, se vi sono più argomenti, l'ordine in cui digitarli.

Questo *ensemble* sintattico/semantico caratterizza non solo LaTeX, ma i linguaggi informatici di alto livello.

Poiché l'autore del pacchetto o della classe è libero di cambiarne l'implementazione pur mantenendo il più possibile inalterate la sintassi e la semantica a cui si riferiscono gli utenti, in definitiva il linguaggio può esser visto come l'interfaccia pubblica del codice.

Ogni comando converrà che sia progettato per esporre un nome breve che sinteticamente ricordi l'elaborazione che sarà effettuata e per ordinare, secondo una logica intuitiva, la sequenza degli argomenti necessari. Di più, converrà che l'intero set di comandi adotti una "filosofia linguistica" uniforme così da facilitarne l'apprendimento e l'uso da parte degli utenti che gestiranno quindi un unico concetto linguistico.

Questo è particolarmente importante se si tiene conto che LaTeX è l'opposto di un linguaggio statico. Pacchetti e nuove classi possono estendere le funzionalità compositive, pertanto molti nuovi elementi del linguaggio vengono resi disponibili nei singoli documenti sorgenti.

Tenteremo di applicare correttamente i principi di progettazione del linguaggio per definire la sintassi dell'ambiente di esempio, e di raggiungere l'equilibrio tra il contenuto informativo puramente testuale del sorgente LaTeX e la restituzione nel documento finale, tipico dell'eleganza dei sistemi asincroni di editoria digitale.

1.1 Un esempio semantico

Per comporre testo sulla pagina si può usare il comando `\parbox`, che prevede due argomenti obbligatori: la larghezza del paragrafo ed il testo.

1. Tralasciamo per un attimo la struttura del testo sorgente e gli ambienti.

Tabella 1: Una semplice tabella d'esempio

Previsione spese d'impianto

	Descrizione voce	€
A	Fornitura principale	80 000,00
B	Costi gestionali	45 000,00
	Costo impiantistico (A+B)	**125 000,00**
C	IVA (20,00% su A+B)	25 000,00
	Totale complessivo (A+B+C)	**150 000,00**

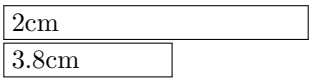
È naturale pensare che i due argomenti debbano essere ordinati nella sequenza sopra citata ed in effetti la sintassi lo conferma:

`\parbox[`⟨*position*⟩`]{`⟨*width*⟩`}{`⟨*text*⟩`}`

Fortunatamente non è usuale che il *testo* del paragrafo sia una misura espressa nelle unità comprese da TEX, dunque ci si accorgerebbe immediatamente dell'errore nel documento finale, come invece potrebbe non accadere nel seguente esempio:

```
\parbox{3.8cm}{2cm}
\parbox{2cm}{3.8cm}
```

che produce (la riquadratura è aggiunta per migliorare la comprensione):



2 Il nuovo ambiente calctab

Spesso mi capita di dover compilare una tabella di liquidazione in documenti aziendali (come quella mostrata in Tabella 1).

Tali tabelle riportano un testo descrittivo ed alcune righe in cui compaiono voci economiche come spese, imposte e somme.

Per costruirle nel sorgente LATEX occorre definire un normale ambiente tabular, digitare i giusti comandi per comporre le righe di separazione, calcolare i valori numerici e digitarli a mano nel giusto formato, digitare le descrizioni delle voci, risistemare il sorgente per allineare le colonne più chiaramente, ecc.

E se cambia un valore occorre procedere al ricalcolo manuale ed alla correzione delle cifre.

L'idea è di sviluppare un nuovo ambiente con cui esprimere i dati essenziali direttamente riferiti alle voci economiche, e di lasciare che il codice, dietro le quinte, si occupi sia dei dettagli grafici sia dei calcoli.

Per i diversi tipi di voci economiche si è pensato di associarvi un comando dedicato:

- `\amount` – inserimento voci economiche
- `\perc` – inserimento voci a percentuale
- `\add` – calcolo di somme

Ai comandi dovranno essere passati due argomenti obbligatori: ⟨*description*⟩ e ⟨*value*⟩ in ordine di lettura nel rigo di tabella, tranne per `\add` che non ha bisogno di un valore poiché ottenuto dall'elaborazione. L'argomento ⟨*value*⟩ per il comando `\perc`, sarà interpretato come la percentuale da applicare.

Per generare la Tabella 1 basta scrivere il seguente codice in cui per riferimento, nei commenti a fine riga sono indicate le lettere che verranno assegnate ai righi in tabella:

```
\begin{calctab}[Previsione spese d'impianto]
  \amount{Fornitura principale}{80000}  % A
  \amount{Costi gestionali}{45000}      % B
  \add{Costo impiantistico}
  \perc{IVA}{20}                        % C
  \add{Totale complessivo}
\end{calctab}
```

Regole generali dell'ambiente: ciascun comando genera una riga di tabella. Le operazioni di calcolo richieste vengono effettuate solo sui valori delle righe precedenti ad esclusione di quelli inseriti con il comando `\add`.

Così nell'esempio di Tab. 1 la percentuale del 20% (rigo C) sarà applicata ai valori espressi nelle precedenti righe (A e B), mentre l'ultimo comando sommerà i valori inseriti dai comandi `\amount` e `\perc` relativi ai righi A, B, e C.

I tre nuovi comandi accettano anche un argomento opzionale (dipendente dal comando) che può essere utilizzato per assegnare un nome al rigo generato dal comando stesso, e/o specificare una lista di nomi tra quelli già assegnati a cui limitare l'elaborazione.

I nomi identificativi di riga possono essere scelti dall'utente in modo significativo così da renderne più naturale l'utilizzo, ed attribuiti con sintassi in stile *chiave* = *valore*. La chiave scelta è semplicemente `id`.

La selezione opzionale dei righi e quindi anche dei valori corrispondenti, si costruisce invece inserendo l'elenco dei relativi nomi separati da virgole nell'argomento opzionale dei comandi.

Di seguito compare il codice d'esempio di un ambiente calctab in cui si etichettano alcune righe allo scopo di applicare diverse aliquote d'imposta (il risultato è la Tabella 2):

Tabella 2: Una seconda tabella d'esempio

Revisione spese d'impianto

	Descrizione voce	€
A	Fornitura	82 500,00
B	Opere strutturali	28 000,00
C	Gestione commessa	46 500,00
D	IVA (10,00% su A+B)	11 050,00
E	IVA (20,00% su C)	9 300,00
	Totale IVA (D+E)	**20 350,00**
	Totale complessivo (A+B+C+D+E)	**177 350,00**

```
\begin{calctab}[Revisione spese d'impianto]
  \amount[id=impianto]{Fornitura}{82500}
  \amount[id=lavori]{Opere strutturali}{28000}
  \amount[id=gest]{Gestione commessa}{46500}
  \perc[id=iva10,impianto,lavori]{IVA}{10}
  \perc[id=iva20,gest]{IVA}{20}
  \add[iva10,iva20]{Totale IVA}
  \add{Totale complessivo}
\end{calctab}
```

Si noti come in questo caso, sia stata data al comando `\perc`, sia l'etichetta (`iva10`), sia la lista di righi precedenti su cui operare (`impianto,lavori`), ed è l'unico comando che accetta entrambe le cose.

Il comando `\add` non può essere etichettato[2], mentre al comando `\amount` non può essere passata una lista di identificatori costituendo una voce economica sempre indipendente dalle altre.

2.1 Specifica di sintassi

La sintassi dell'ambiente calctab è la seguente:

```
\begin{calctab}[⟨description paragraph⟩]
    ⟨calctab commands list⟩
\end{calctab}
```

dove per ⟨*calctab commands list*⟩ si intende una sequenza dei comandi:

```
\amount[id=⟨rowID⟩]{⟨descr⟩}{⟨value⟩}
```

```
\perc[id=⟨rowID⟩,⟨rowID list⟩]{⟨descr⟩}{⟨value⟩}
```

```
\add[⟨rowID list⟩]]{⟨descr⟩}
```

3 Test d'utilizzo

Ho chiesto ad alcuni colleghi di scrivere una classica tabella di liquidazione di una notula professionale avvalendosi di un ambiente calctab. Un compito che essi svolgono quasi quotidianamente, dunque un tema di loro interesse, che normalmente viene svolto con l'uso di comuni word processor o fogli di calcolo.

2. Questa scelta mantiene più chiara la sintassi a scapito di un po' di funzionalità. Vedi la sezione 3 per la discussione in merito.

Durante colloqui informali individuali, ho mostrato loro dapprima alcuni esempi di sorgenti compilabili per abituarli al linguaggio di markup (nessuno infatti, aveva mai nemmeno sentito nominare la parola LaTeX), infine ho fornito loro una tabella d'esempio cedendogli la tastiera. Ho diligentemente registrato le osservazioni e le domande che durante il test ciascun collega mi poneva nell'intento di misurare la *soddisfazione del cliente*. Gli incontri hanno confermato l'utilità di discussioni simili al fine di migliorare con efficacia il *prodotto* (il lavoro di squadra spesso è insuperabile).

La maggior parte delle domande che mi sono state rivolte riguardanti l'ambiente calctab, erano di fatto richieste di estensioni. Ad esempio, la possibilità di cambiare "unità di misura" per i valori in tabella (oltre l'euro), e la gestione di nuove colonne *Quantità* e *Prezzo* tipiche di un prospetto economico in cui il valore di riga corrisponde al prodotto di tali numeri. Od ancora la possibilità di mutare le lettere di riga in prima colonna, in numeri progressivi (vedi le Tabelle 1 e 2).

A quest'ultima richiesta ho obiettato che le lettere maiuscole risultano più leggibili nelle lunghe espressioni di descrizione delle somme: il richiamo (A+B+C+D+E), visibile nell'ultima riga in Tab. 2 è forse più chiaro di (1+2+3+4+5).

Sull'argomento più importante, la sintassi di calctab, sono stati dibattuti due temi, entrambi inerenti alle etichette opzionali dei comandi.

Il primo verte sulla possibilità di ricorrere ad un'etichettatura automatica anziché esplicita delle righe di tabella, per esempio con la sequenza `row1`, `row2`, ..., dunque per comporre la tabella 2 avremmo dovuto scrivere il codice seguente:

```
% modifica proposta alla sintassi:
\begin{calctab}[Revisione spese d'impianto]
  \amount{Fornitura}{82500}              % 1
  \amount{Opere strutturali}{28000}      % 2
  \amount{Gestione commessa}{46500}      % 3
  \perc[row1,row2]{IVA}{10}              % 4
  \perc[row3]{IVA}{20}                   % 5
  \add[row4,row5]{Totale IVA}
  \add{Totale complessivo}
\end{calctab}
```

Nell'approfondito dibattito che ne è seguito, ho giustificato la diversa scelta adottata, proponendo i seguenti argomenti senz'altro opinabili: in primo luogo ci si deve ricordare i caratteri utilizzati per costruire la sequenza delle etichette, e ciò è più faticoso che non ricordarsi il nome della chiave `id`.

In secondo luogo occorre *contare* le righe inserite per formarne effettivamente i nomi e questo è meno elegante ed espressivo del metodo di assegnare direttamente nomi consistenti alle righe. Infine, in caso di modifica od inserimento di nuove righe occorre che sia aggiornato manualmente l'elenco dei parametri opzionali già digitati.

Il secondo tema invece, sembra una buona idea: poter assegnare un identificatore di riga nell'argomento opzionale del comando `\add`. In altri termini, si tratta di poter assegnare un nome anche alle righe di somma di valori, sia essa totale o selettiva, per compiervi ulteriori elaborazioni senza dover identificare ciascun rigo "addendo" per poi comporne l'elenco nel comando.

In questo modo una tabella di notula professionale come la Tab. 3 (pag. 35) si risolverebbe con l'assegnamento di un unica chiave `id`. In particolare basterebbe assegnare l'id al comando che somma le voci di notula (A+B+C) e dichiararlo come lista nel comando che effettua il calcolo della ritenuta d'acconto al rigo F.

Evitare di etichettare tutte le righe solo per riottenerne la somma per righi successivi è uno strumento agile che completa il linguaggio. Tale valore infatti, è già disponibile nella riga di somma ed è naturale che si possa richiamarlo da qualche riga più sotto.

La scelta sintattica iniziale è risultata quindi migliorabile sia dal punto di vista concettuale che funzionale. Il codice dovrà essere modificato con l'aggiunta di un nuovo *stack* di registri in virgola mobile con i relativi *flag booleani* di stato. In questo modo i valori numerici principali inseriti dai comandi `\amount` e `\perc` rimangono indipendenti e le regole base dell'ambiente ancora valide.

4 Implementare il linguaggio

L'implementazione del nuovo ambiente calctab è stata sviluppata per stadi, risolvendo un problema alla volta e facendo in caso ricorso, a post sul forum di GUIT come accennato nella sezione 8.

La strategia di programmazione si è inoltre orientata verso l'uso di pacchetti LaTeX già disponibili, visti un po' come librerie di codice.

4.1 Eseguire calcoli

Per calcoli con numeri interi, TeX dispone dei contatori `\count`, numeri con segno a 32 bit il cui range è $[-2^{31}, 2^{31}-1]$. Tra l'altro il limite superiore dell'intervallo fissa a poco più di due miliardi il numero massimo di pagine di un documento (vedi Gregorio (2008) pagine 35 e 42).

Per i numeri decimali si possono impiegare registri dimensionali, per esempio `\dimen`, dalla notazione in *virgola fissa* con 14 bit per la parte intera, 16 bit per quella frazionaria, un bit di segno ed un bit di overflow, per un totale di ancora 32 bit (l'intervallo possibile è $[-2^{14}, 2^{14}-1]$).

Il limite superiore stabilisce la massima lunghezza a circa 5,76 m ($\approx 2^{14} = 16.384$ pt)[3], mentre la precisione raggiungibile è 2^{-16} pt $\approx$ 5,36 nm, lunghezza che in TeX viene definita punto scalato `sp`. Il valore è assai meno della lunghezza d'onda della luce visibile compreso tra 380 e 750 nanometri, quindi errori dovuti ad approssimazioni numeriche sono irrilevabili nel documento.

Può essere interessante notare che nel modulo pgfmath, il motore matematico del noto pacchetto grafico PGF, i calcoli decimali sono svolti con l'ausilio dei registri dimensionali TeX, come dichiara il manuale alla sezione 49 (Tantau, 2008).

Se da un lato questi "tipi" numerici bastano per le esigenze di TeX, dall'altro all'epoca del suo sviluppo l'aritmetica in *virgola mobile* non garantiva una sufficiente congruità tra le diverse piattaforme hardware e software, così che Donald Knuth non la implementò per TeX.

In effetti, a detta dello stesso Knuth, il problema del calcolo in *virgola mobile* è più complesso e sorprendente di quello che si potrebbe pensare (Knuth, 1998). Esso coinvolge aspetti teorici, di progettazione dei processori e di sviluppo di standard internazionali[4] (Kahan, 1981).

Grazie a più di vent'anni di evoluzione di questo importante settore del calcolo, future incarnazioni di TeX potranno disporre di questo tipo di aritmetica per altro utile in molti tipi di elaborazioni di tipocomposizione (Beebe, 2007).

Ritornando all'ambiente calctab, questo necessita di operare con numeri decimali e con valori che superano i limiti dei registri dimensionali.

Per fortuna sono stati implementati pacchetti LaTeX che svolgono operazioni in *virgola mobile* anche se in maniera non efficiente. Si è optato per fltpoint (Guthöhrlein, 2004) che offre la possibilità dell'uso di comodi registri, impiegati per la memorizzazione dei valori di riga di calctab. Il codice della tabella è così in grado di eseguire le operazioni previste con la necessaria precisione e facilmente supportarne altre future.

4.2 I pacchetti di "libreria"

Fondamentale per la semplicità d'implementazione il pacchetto xkeyval (Adriaens, 2006), con cui non solo si è risolto il problema posto dalla prevista sintassi dell'argomento opzionale dei comandi interni di calctab, ma si è ottenuto una notevole eleganza e

3. 1 pt $= \frac{25,4}{72,27}$ mm $\approx$ 0,35 mm e 1 nm $= 10^{-9}$ m.

4. Lo standard IEEE 754 – "Standard for Binary Floating-Point Arithmetic" del 1985 fu il primo punto di riferimento in materia.

compatezza del codice che lo utilizza (vedi sezione 4.3).

Per procedere all'implementazione degli algoritmi di calcolo, ci si è avvalsi del potente ifthen (Carlisle, 2001), su cui è basato il loop di calcolo principale (comune sia ai comandi `\perc` e `\add`), nonché utilizzato per costruire una serie di flag di controllo.

La formattazione delle cifre a due decimali fissi, è stata invece affidata al pacchetto numprint (Harders, 2008), che ci si aspetta renda agevoli ulteriori miglioramenti e sviluppi dell'ambiente calctab.

Si è invece scartato (solo per il momento) l'uso del pacchetto datatool (Talbot, 2007) per difficoltà incontrate nella composizione grafica della tabella, problema la cui soluzione probabilmente richedeva l'esperienza dell'averlo risolto con gli strumenti di base. Si tratta comunque di un pacchetto potente che gestisce tabelle di dati in maniera simile a quelle dei database.

I pacchetti utilizzati per l'aspetto grafico della tabella sono stati: booktabs per l'ovvio miglioramento dei filetti orizzontali, eurosym per il simbolo dell'euro, ed infine xcolor con l'opzione table utilizzato per colorare alternativamente le righe della tabella.

4.3 Codice delle chiavi

La definizione di una chiave `id` (ordinaria secondo la classificazione in xkeyval), relativa all'etichetta di riga comporta la creazione di una seconda chiave il cui nome è il valore stesso dell'`id`.

Nel codice riportato di seguito [5] [6] si noti come il parametro `#1` venga espanso a formare il nome del registro numero di riga.

```
% the ordinary key is
% useful to assign a name
% to the table rows
% Note the use of different families

\define@key{ctfamId}{id}{
  % save its row number in ctreg<idname>
  \fpRegSet{ctreg#1}{\thectRowIndex}
  \define@key{ctfamLabel}{#1}[]{
    %load the corresponding row number
    \fpRegGet{ctreg#1}{\tempnumrow}
    % activate ctflag<row>
    \setboolean{ctflag\tempnumrow}{true}
    % global flag is false if a key
    % is specifed
    \setboolean{ctGlobflag}{false}
  }
}
```

5. Le linee sono state accorciate per non andare oltre la larghezza di colonna.

6. Il codice va racchiuso tra i comandi `\makeatletter` e `\makeatother` se compare nel preambolo del file sorgente e non in un file di pacchetto `.sty`.

Per memorizzare il numero di riga e risolvere l'associazione con la chiave `id`, si rende necessario creare un registro fltpoint per ciascuna di essa[7].

Quando una chiave compare nella ⟨*rowID list*⟩, deve essere attivato il flag che indica che la riga corrispondente dovrà essere considerata nel calcolo, mentre lo stato del flag globale dovrà essere `false` ad indicare un calcolo selettivo.

In altre parole, il flag globale `ctGlobflag` indica lo stato del calcolo: il `true` indica che il calcolo dovrà riguardare tutti i valori già introdotti in tabella (con l'ausilio di comandi `\amount` e `\perc`), il `false` indica che solo le righe i cui `ctflag<row>` sono `true` saranno elaborate nel calcolo.

4.4 Codice dei comandi interni

L'esecuzione del calcolo viene portata a termine con un comando privato, interno all'ambiente, che restituisce il risultato dell'elaborazione nel registro `ctSum`, e nella stringa descrittiva memorizzata in `\ctlabelreg`, quest'ultima utilizzata in *append* su se stessa, effettuando un ciclo per ciascuna riga.

L'importante comando `\ctLoop` utilizza il costrutto `\whiledo` contenente l'istruzione condizionale `\ifthenelse`, come si può notare nel seguente listato:

```
\newcommand\ctLoop{
\fpRegSet{ctSum}{0}
\setcounter{ct}{1}
\edef\ctlabelreg{}% for text of label rows
\edef\ctplustext{}% for '+' char
\whiledo
{\not\(\value{ct}>\value{ctRowIndex}\)}{
  \ifthenelse
    {\(\boolean{ctflag\thect}%
            \OR%
       \boolean{ctGlobflag}\)}
    {\fpRegAdd{ctSum}{ctRowValue\thect}
    % row flag status reset
    \setboolean{ctflag\thect}{false}
    \edef\ctlabelreg
      {\ctlabelreg\ctplustext\Alph{ct}}
    \edef\ctplustext{+}
    }{}% else clause denied
  \stepcounter{ct}%
}%
  % reset of the global flag status
  \setboolean{ctGlobflag}{true}
}%
```

Il contatore LaTeX chiamato `ctRowIndex` detiene il numero di righe fino a quel momento inserite.

Per convertire il numero di riga in lettere si è fatto uso della funzione del kernel di LaTeX `\Alph` che gestisce l'intervallo da 1 a 26 (eventualmente esiste il pacchetto alphalph di Heiko Oberdiek (Oberdiek, 2007) che estende oltre tale limite la traduzione in lettere dei numeri interi utilizzando più caratteri alfabetici).

7. Occorre migliore l'implementazione sostituendo l'uso dei registri fltpoint.

Ad ogni ciclo viene "spento" il flag di riga eventualmente attivo per effetto del codice relativo alle chiavi di lista, e, all'uscita, si riattiva comunque il flag globale `ctGlobflag` riportandolo nello stato corrispondente all'elaborazione di tutte le righe.

Poiché i comandi di riga sono definiti internamente all'ambiente calctab (usarli all'esterno non avrebbero senso), nel seguente codice del comando `\amount`, gli argomenti hanno un doppio `#`.

```
\newcommand\amount[3][]{
  \stepcounter{ctRowIndex}%          new row
  % definition of the bool flag for each row
  \newboolean{ctflag\thectRowIndex}
  % starting value of ctflag<n>
  \setboolean{ctflag\thectRowIndex}{false}
  \setkeys{ctfamId}{##1}
  % save the amount value
  \fpRegSet{ctRowValue\thectRowIndex}{##3}
  % append the row to the token register
  \toks255=\expandafter{\the\toks255\Alph{%
    ctRowIndex}&##2\rule{\ctsep}{0pt}&
    \numprint{##3}\\}
  }
```

È stato previsto uno spazio orizzontale regolabile, fissato a 8 mm, detto `\ctsep`. Tale spazio viene inserito in seconda colonna per impedire che in alcuni casi la tabella possa apparire troppo stretta.

In effetti, l'output dei comandi interni consiste in una sequenza di token memorizzati nel registro TEX `\toks255`. Questo permette di risolvere il problema conseguente al fatto che il contenuto di ciascuna cella della tabella composta dall'ambiente tabular, viene racchiuso in un gruppo implicito. Tale oggetto, descritto per esempio in Gregorio (2008) al Capitolo 2, costituisce una barriera per le definizioni non globali, interne all'ambiente.

Il contenuto del registro verrà poi "liberato" solo dal codice di chiusura dell'ambiente calctab.

Il comando `\add` elabora la lista di chiavi e poi lancia il ciclo che renderà disponibile il valore della somma nel registro `ctsum`. Dopodiché non dovrà far altro che aggiungere i token al registro `\toks255` che alla fine conterrà l'intera tabella:

```
\newcommand\add[2][]{%
  \setkeys{ctfamLabel}{##1}
  \ctLoop
  \fpRegGet{ctSum}{\atempnumber}
  \toks255=\expandafter{\the\toks255%
    \midrule\rowcolor{gray!15}&%
    \bfseries##2 (}
  \edef\ctnum{\noexpand\ctlabelreg)%
    noexpand\rule{\ctsep}{0pt}}
  \toks255=\expandafter\expandafter%
    \expandafter{\expandafter\the%
    \expandafter\toks255\ctnum&\bfseries}
  \edef\ctnum{\noexpand%
    \numprint{\atempnumber}}
  \toks255=\expandafter\expandafter%
    \expandafter{\expandafter\the%
    \expandafter\toks255\ctnum\\}
  }
```

Nelle prossime linee di codice è riportato il comando `\perc`, quello più complesso dei tre.

Si noti come, prima di lanciare il ciclo `\ctLoop`, deve essere momentaneamente decrementato il contatore di riga per evitare che la riga stessa sia conteggiata nel calcolo.

Dopodiché viene richiesta l'elaborazione delle chiavi: prima quelle di lista (famiglia `ctfamLabel`), poi quella di etichettatura (famiglia `ctfamId`), onde evitare errori nel caso in cui l'utente assegni il nome di riga ed allo stesso tempo lo inserisca nella lista di etichette.

Vengono poi eseguiti i calcoli in virgola mobile per applicare la percentuale indicata dall'utente alla somma contenuta, dopo l'esecuzione di `\ctLoop`, nel registro fltpoint `ctsum`.

Infine è la volta dei comandi di aggiornamento del registro `\toks255`.

```
\newcommand\perc[3][]{
  % another new row:
  \stepcounter{ctRowIndex}
  % a control bool flag for each row:
  \newboolean{ctflag\thectRowIndex}
  % starting value of ctflag<n>:
  \setboolean{ctflag\thectRowIndex}{false}
  % processing of the user id label:
  \setkeys{ctfamLabel,ctfamId}{##1}
  % get the value of the sum:
  \addtocounter{ctRowIndex}{-1}
  \ctLoop
  \stepcounter{ctRowIndex}

  % calculation
  \fpDiv{\tempPerc}{##3}{100}%
  \fpRegSet{tempreg}{\tempPerc}
  \fpRegMul{ctSum}{tempreg}
  \fpRegRound{ctSum}{-2}%
  \fpRegGet{ctSum}{\atempnumber}
  \fpRegSet{ctRowValue\thectRowIndex}
     {\atempnumber}
  \toks255=\expandafter{\the\toks255%
     \Alph{ctRowIndex}&##2 %
     (\numprint{##3}\% su }
  \edef\ctnum{\noexpand\ctlabelreg)%
     \noexpand\rule{\ctsep}{0pt}}
  \toks255=\expandafter\expandafter%
     \expandafter{\expandafter\the%
     \expandafter\toks255\ctnum&}
  \edef\ctnum{\noexpand\numprint{%
     \atempnumber}}
  \toks255=\expandafter\expandafter%
     \expandafter{\expandafter\the%
     \expandafter\toks255\ctnum\\
  }
}
```

4.5 Codice dell'ambiente calctab

Occorre inizialmente allocare contatori, lunghezze e box. Il codice di chiusura del nuovo ambiente calctab sistema i token del registro, e quindi la tabella, in un box per misurarne la larghezza con il comando primitivo `\wd`.

Tale dimensione diventerà la larghezza del materiale di intestazione della tabella, passato all'ambiente come argomento opzionale.

Ecco il listato:

```
% length for tuning width
% of the central column:
\newlength{\ctsep}\setlength{\ctsep}{0.8cm}

% main row counter of table:
\newcounter{ctRowIndex}
% loop index:
\newcounter{ct}

% table box allocation
\newsavebox{\ctTabBox}

\newenvironment{calctab}[1][]{%
  % set font family
  \sffamily

  % set the token register
  \toks255={\rowcolors{1}{gray!15}{}%
    \begin{tabular}{clr}\midrule&%
    Descrizione voce & \euro \\\midrule}

  % reset counter to starting value
  \setcounter{ctRowIndex}{0}

  % save paragraph table header to use it
  % in the environment end code
  \newcommand\ctFirstPar[1]{#1}

  % set standard decimal position
  \nprounddigits{2}

  % new boolean to control numeric loop
  \newboolean{ctGlobflag}
  \setboolean{ctGlobflag}{true}

  % insert \ctLoop code ...
  % insert \amount code ...
  % insert \perc code ...
  % insert \add code ...

}{%closing code
\begin{center}
  \begin{lrbox}{\ctTabBox}
  \the\toks255\bottomrule
  \end{tabular}
  \end{lrbox}
  \sffamily
  \parbox{\wd\ctTabBox}{\ctFirstPar\par%
  \smallskip\par
  \usebox{\ctTabBox}}
\end{center}
}
```

5 Un comando, una tabella

È interessante notare che si possono costruire nuovi comandi per la composizione immediata di tabelle di tipo calctab.

Nei documenti della contabilità dei lavori pubblici, per esempio, può essere necessario riassumere le voci economiche riguardanti l'emissione di uno Stato d'Avanzamento Lavori (SAL) che consistono in una prima riga con l'importo dei lavori, una seconda con l'IVA, ed un ultima con il totale.

Per ottenere la tabella può essere predisposto un nuovo comando, chiamato `\saltab`, a cui passare i tre argomenti numerici in un ordine ritenuto essere il più logico:

1. il numero progressivo del SAL emesso,
2. l'importo dei lavori,
3. l'aliquota IVA da applicare.

Il nuovo comando `\saltab` ha dunque la sintassi seguente:

`\saltab{`⟨*SAL number*⟩`}{`⟨*amount*⟩`}{`⟨*rate*⟩`}`

e questa semplice implementazione:

```
\newcounter{cttemp}
\newcommand\saltab[3]{% tabella per SAL
  \setcounter{cttemp}{#1}
  \begin{calctab}[Tabella liquidazione %
                  del \Roman{cttemp} %
            Stato d'avanzamento Lavori]
    \amount{Lavori}{#2}
    \perc{IVA}{#3}
    \add{Totale liquidato}
  \end{calctab}}
```

6 L'esempio classico

L'ultimo esempio riguarda la tabella 3 in cui è necessario calcolare una percentuale con due cifre decimali significative ed apporre un testo descrittivo su più linee. È il classico caso della liquidazione di una notula professionale.

7 Conclusioni

Il lavoro svolto ha consentito la messa a punto di una sintassi semplice e l'implementazione di un nuovo ambiente tabella in grado di effettuare calcoli.

Ciò dimostra l'utilità di LaTeX anche in ambiti aziendali dove può anche superare in produttività i software di Office Automation WYSIWYG.

La maggior diffusione di LaTeX dipende molto dal modo di interagire con il linguaggio.

Comandi intuitivi ed ambienti dal giusto insieme di opzioni, nonché nuovi pacchetti o classi di documento personalizzati, consentono all'utente di dedicarsi ai contenuti, e produrre documenti con minori errori e qualità tipografica elevata.

Tuttavia, senza un forte impegno e motivazione da parte degli utenti, LaTeX è destinato ai margini delle scelte IT sopratutto delle imprese, nonostante la gratuità e la libertà concessa dalla licenza LPPL.

La risposta dei colleghi che hanno compiuto le prove mi induce a concludere che l'investimento

Tabella 3: Esempio classico: la notula professionale

```
\begin{calctab}[Notula per prestazioni professionali:\\
                Progetto impiantistico per la pizzeria ``La Margherita'']
    \amount[id=aria]{Progettazione impianto trattamento aria}{5400}
    \amount[id=video]{Progettazione impianto elettrico e di sorveglianza video}{8000}
    \perc[id=spese]{Spese generali}{8,55}
    \add{Importo totale}
    \perc{Contributo cassa nazionale}{2}
    \perc{IVA}{20}
    \perc[aria,video,spese]{Ritenuta d'acconto}{-20}
    \add{Totale complessivo}
\end{calctab}
```

Notula per prestazioni professionali:
Progetto impiantistico per la pizzeria "La Margherita"

	Descrizione voce	€
A	Progettazione impianto trattamento aria	5 400,00
B	Progettazione impianto elettrico e di sorveglianza video	8 000,00
C	Spese generali (8,55% su A+B)	1 145,70
	Importo totale (A+B+C)	**14 545,70**
D	Contributo cassa nazionale (2,00% su A+B+C)	290,91
E	IVA (20,00% su A+B+C+D)	2 967,32
F	Ritenuta d'acconto (−20,00% su A+B+C)	−2 909,14
	Totale complessivo (A+B+C+D+E+F)	**14 894,79**

iniziale in risorse e cultura informatica da mettere in campo per il proficuo uso di LaTeX sia notevole, ma il paradigma offerto dalla sintassi elegante ed espressiva del linguaggio crea forse uno strumento migliore con cui lavorare.

8 Ringraziamenti

Ringrazio in primo luogo il Professor Enrico Gregorio per avermi aiutato, per mezzo del forum di GuIt, a risolvere il problema costituito dai gruppi impliciti in cui sono racchiuse le celle dell'ambiente tabular, per avermi suggerito il modo di uguagliare la larghezza del `\parbox` d'intestazione a quella della tabella, e per avermi guidato nella *giungla* degli `\expandafter`.

Un ringraziamento va anche ai miei colleghi, che si sono sottoposti ad una "quindicina di minuti di perplessità" durante i test sintattici, all'amica Marzia Dati per l'aiuto con la lingua inglese, ed alla mia famiglia, sempre felice di concedermi tempo e comunicarmi incoraggiamento per i progetti come questo.

Riferimenti bibliografici

Adriaens, H. (2006). «The xkeyval package». Disponibile su CTAN, `http://www.ctan.org/tex-archive/macros/latex/contrib/xkeyval`.

Beebe, N. H. F. (2007). «Extending TeX and METAFONT with floating-point arithmetic». In *The 28th Annual Meeting of the TeX Users Group*. Testo riassuntivo in `http://www.math.utah.edu/~beebe/talks/2007/tug2007/tug2007.pdf`.

Carlisle, D. (2001). «The ifthen package». Disponibile su CTAN, `http://www.ctan.org/tex-archive/macros/latex/base`.

Gregorio, E. (2008). «Appunti di programmazione in LaTeX e TeX». `http://profs.sci.univr.it/~gregorio/introtex.pdf`.

Guthöhrlein, E. (2004). «The fltpoint package». Disponibile su CTAN, `http://www.ctan.org/tex-archive/macros/latex/contrib/fltpoint`.

Harders, H. (2008). «The numprint package». Disponibile su CTAN, `http://www.ctan.org/tex-archive/macros/latex/contrib/numprint`.

Kahan, W. (1981). «Why do we need a floating-point arithmetic standard?» Scaricabile da `http://www.cs.berkeley.edu/~wkahan/ieee754status/why-ieee.pdf`.

Knuth, D. E. (1998). *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley Professional.

Oberdiek, H. (2007). «The alphalph package». Disponibile su CTAN, `http://www.ctan.org/tex-archive/macros/latex/contrib/oberdiek/alphalph.pdf`.

Talbot, N. (2007). «The datatool package». Disponibile su CTAN, http://www.ctan.org/tex-archive/macros/latex/contrib/datatool.

Tantau, T. (2008). «The Ti*k*Z and pgf packages». Scaricabile da http://www.ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf. Manual for version 2.00.

▷ Roberto Giacomelli
Carrara (MS), Italia
giaconet at tin dot it