

Macroistruzioni con argomenti delimitati

Claudio Beccari

Sommario

Il pacchetto di macroistruzioni comunemente definito \LaTeX non tratta mai delle macro con argomenti delimitati; esso stesso ne mette a disposizione del compositore un numero minimo. Con i comandi primitivi dell'interprete \TeX è possibile definire macro con argomenti delimitati che si possono usare anche con \LaTeX . Queste macro possono risultare estremamente comode in certe circostanze, specialmente quando si creano file di classe o pacchetti di estensione, dove permettono di identificare con maggiore facilità il significato di ogni argomento. L'esposizione viene eseguita servendosi di un problema concreto: il Registro delle Lezioni.

Abstract

The macro package commonly known as \LaTeX does not describe any means for defining macros with delimited arguments; furthermore it offers a very small number of them to the user. By means of the primitive commands of the \TeX interpreter it is easy to define macros with delimited arguments that may be used also while using the other \LaTeX macros. This kind of macros may be very useful in some instances, particularly when writing class or package files, where they make it easy to identify the function that any argument plays in the macro expansion. The subject is described with the aid of a practical problem: the Lecture Log.

1 Introduzione

Le macroistruzioni, comunemente chiamate 'macro', servono per far eseguire all'interprete tex una serie complessa di operazioni, che il compositore non è costretto a riscrivere ogni volta per disteso.

L'interprete, che sia tex stesso o pdfetex o altro, è un programma eseguibile che interpreta quanto è scritto nel file sorgente .tex , per eseguire la composizione del testo che vi è contenuto insieme ai comandi di *mark-up* che dichiarano certe parti del testo come appartenenti a certe categorie, che vanno composte in questa o quella maniera.

L'interprete nella totalità dei casi non costringe il compositore a scrivere il testo inframmezzato ai comandi di composizione, siano essi cambiamenti di font, variazioni della giustezza o della giustificazione, passaggio alla composizione della matematica, eccetera; anche la semplicissima collezione di macro contenuta nel file plain.tex permette al compositore di usare un set essenziale di macro

che consentono di arrivare ad un modesto risultato senza troppa fatica¹. La collezione di macro contenuta nel file latex.ltx mette a disposizione del compositore un numero sterminato di macro che agevolano ulteriormente la composizione, lasciando l'autore e il compositore liberi di concentrarsi sul contenuto, piuttosto che sulla sua estetica. Le centinaia di pacchetti di estensione disponibili negli archivi internazionali estendono ulteriormente le funzionalità delle macro di \LaTeX .

Lo stesso succede quando si usano altri programmi di composizione quali Omega (Ω) e Lambda, oppure Aleph e Lamed, oppure Con \TeX t o $X_{\text{q}}\LaTeX$ e $X_{\text{q}}\LaTeX$.

Per rendere più spedito l'avvio della compilazione, fin dall'inizio \TeX è stato progettato con la capacità di leggere una volta per tutte il file plain.tex , o il file latex.ltx , o altri simili file, trasformandone il contenuto in linguaggio macchina, per poi scaricare la parte di memoria occupata da questa trasformazione in un file, detto di 'formato', nella fattispecie plain.fmt e latex.fmt , così che in seguito è più veloce caricare il file binario di formato e si risparmia tutto il lavoro di interpretazione e di trasformazione in linguaggio macchina già fatto una volta per tutte. Quando si lancia \LaTeX direttamente dalla linea di comando o cliccando su di un opportuno bottone dello *shell editor*, in realtà si dà il comando di lanciare l'interprete dicendogli anche quale file di formato usare, in questo caso il formato latex.fmt .

Quando è in azione, l'interprete legge il flusso di dati che gli arriva dal file sorgente, e assegna una categoria ad ogni oggetto (*token*) che legge; poi, a seconda della categoria, esegue quanto ogni token dice di fare; se si tratta di una semplice lettera, la accoda al testo precedentemente raccolto, se si tratta di un segno di interpunzione lo inserisce nel testo con le opportune spaziature, eccetera.

Ma se si tratta di un comando, esamina se si tratta di un vero comando primitivo, nel qual caso lo esegue, altrimenti si tratta di una macro: in questo caso ne va a cercare la definizione (nel formato caricato in memoria oppure nella parte di memoria dove ha raccolto le macro definite dall'utente o

1. Qui non si discutono i pregi e i difetti di questa o quella collezione di macro; gli estimatori di $\text{plain } \TeX$ hanno dei buoni motivi per preferire questa collezione minimale di macro, perché essa consente loro di specificare quello che vogliono per le parti della composizione che già non hanno una macro definita. Il fatto è che per comporre bene $\text{plain } \TeX$ richiede una enorme esperienza e non poca attenzione; in parole povere un gran lavoro e quindi una notevole fatica.

lette dai pacchetti di estensione), cerca anche se gli argomenti sono delimitati o non lo sono, ed in ogni caso ‘tokenizza’ gli argomenti tratti dal file di ingresso, esegue le formali sostituzioni nei segnaposto di ogni argomento, esamina se la definizione del comando fa ricorso ad altre macro, ripete il processo per ogni altra macro che incontra, finché ha completamente sviluppato la macro iniziale e il suo contenuto ed ha solo una lista di token non ulteriormente sviluppabili; a questo punto esegue in sequenza tutti questi token.

Sembra complicato e lo è; il vantaggio per l’utente del sistema TEX è che questi non deve preoccuparsene, perché l’interprete fa tutto da solo. Al massimo l’utente può trarre giovamento dalla comprensione di questo meccanismo quando incontra degli errori e deve provvedere in merito.

Come si è visto nella sommaria descrizione fatta sopra, l’interprete vede se la definizione di ogni macro prevede argomenti delimitati, oppure se questi argomenti sono semplici token non ulteriormente sviluppabili ovvero sono stringhe di lettere e, talvolta, ulteriori comandi, racchiusi dal compositore dentro una coppia di parentesi graffe, che costituiscono i delimitatori espliciti di default di ciascun argomento. Per esempio, se il compositore vuol comporre una tabella e vi vuole collocare una cella che occupi due colonne con l’argomento costituito da una parola centrata ed evidenziata con il comando `\emph`, inserisce nel file sorgente la dichiarazione

```
\multicolumn2c{\emph{Pippo}}
```

dove i token `2` e `c` non necessitano di parentesi di gruppo perché sono due token non ulteriormente scomponibili, mentre il terzo argomento di `\multicolumn` richiede le parentesi graffe che lo delimitano, così come il terzo argomento stesso contiene un comando con un argomento formato da più lettere, che va quindi racchiuso a sua volta fra parentesi graffe.

Si noti l’ambiguità delle parole che si stanno usando per descrivere la funzione delle parentesi graffe; esse definiscono il loro contenuto come un gruppo, e in un certo senso ne costituiscono i delimitatori; tuttavia esse non delimitano l’argomento, nonostante le apparenze, ma definiscono il gruppo.

La mancanza di delimitatori è ancora più evidente se si va a cercare nel file `latex.ltx` la definizione di `\multicolumn` che, semplificando molto, si riduce a

```
\def\multicolumn#1#2#3{definizione}
```

e, come si vede i tre argomenti `#1`, `#2`, e `#3` non sono separati da nulla, nemmeno da spazi.

2 Macro con argomenti delimitati

Dopo questa lunga introduzione vale la pena di descrivere come, tecnicamente, si possono definire macro con argomenti delimitati.

Innanzitutto non si possono creare simili definizioni usando i comandi di alto livello messi a disposizione da LATEX: `\newcommand` o `\renewcommand` oppure il loro parente stretto `\providecommand` o, infine, `\DeclareRobustCommand` e le loro versioni asteriscate. Questi comandi, infatti, evitano al compositore di doversi preoccupare di molti dettagli e di molte verifiche, e perciò isolano il compositore dalla effettiva definizione eseguita durante l’esecuzione di quei comandi, definizione che alla fine si riduce all’esecuzione di uno dei quattro comandi primitivi dell’interprete TEX: `\def`, `\edef`, `\gdef` e `\xdef`.

Il problema di questi quattro comandi è che sono ‘primitivi’ non solo nel senso che sono all’origine di qualunque altro comando (macro) che possa essere definito tramite loro, ma anche nel senso traslato del termine: nel senso, cioè, che ‘senza ragionare’ eseguono incondizionatamente quello che devono fare, con il rischio che essi vengano usati male dal compositore quando involontariamente ridefinisce un comando del nucleo di LATEX; a causa di questo errore, LATEX smette (o può smettere) di funzionare. Se dovesse succedere una cosa del genere, la caccia all’errore deve essere ricercata per prima cosa nelle definizioni imprudenti fatte dal compositore stesso.

Per noi non anglofoni c’è un metodo molto semplice ed economico per definire comandi diversi da quelli del nucleo di LATEX: basta usare nomi italiani, invece che inglesi, come sono invece tutti i comandi interni di LATEX.

Però il metodo corretto, e valido indipendentemente dalla lingua del compositore, è quello di eseguire la verifica che il nome del comando sia effettivamente disponibile per una definizione: basta scrivere le proprie definizioni all’interno di un file di stile con estensione `.sty`, da leggere nel preambolo del file di ingresso mediante il comando `\usepackage`, oppure far precedere il comando `\makeatletter` alle nuove definizioni, che a loro volta devono essere seguite da `\makeatother`. Volendo per esempio definire la macro `\pipe` mediante il comando `\def`, basta scrivere nel preambolo:

```
\makeatletter
\@ifdefinable{\pipe}{definizione}
\makeatother
```

Il comando `\@ifdefinable`, che contiene nel suo nome la ‘non lettera’ `@`, è il motivo per il quale bisogna premettere `\makeatletter` prima di farne uso, e poi ripristinare `@` con la categoria di ‘non lettera’ dopo averne fatto uso. Per altro, il suddetto comando `\@ifdefinable` controlla che il comando

proposto sia effettivamente non ancora definito, che non cominci per `\end2` e sia diverso da `\relax`, visto che questi due ultimi comandi giocano un ruolo importantissimo nel nucleo di LATEX.

Riferendoci alla predisposizione di una classe per comporre i registri delle lezioni, è possibile che sia necessario separare il nome dal cognome del docente, per associare al cognome un significato particolare; per non costringere il compositore a ricordarsi troppe regole, decidiamo che la stringa di caratteri contenenti il nome e il cognome, che verrà usata più volte nella composizione del registro, sia introdotta dal compositore (verosimilmente il docente stesso) come una stringa contenente un solo spazio di separazione fra nome e cognome³.

Questo unico spazio ci servirà da delimitatore del nome; potremo usare un altro carattere per delimitare il cognome, per esempio un punto esclamativo, sicuramente non presente nel cognome. La nostra definizione sarà composta di due macro; la prima serve al docente per impostare il suo nominativo, che verrà memorizzato in una macro di servizio; il secondo servirà ad isolare il cognome e definirà un'altra variabile temporanea da usare in altre macro, in relazione al significato che vorremo assegnare al cognome. Ecco dunque che avremo:

```
\let\@docente\empty% Default iniziale
\def\docente#1{\def\@docente{#1}}
\def\getcognome#1 #2!{\def\Cognome{#2}}
```

Si noti lo spazio fra il primo e il secondo argomento di `\getcognome` e il punto esclamativo che delimita la fine del cognome.

Ecco, questa è la definizione di una macro ad argomenti delimitati; se si preparasse il seguente file `.tex` e lo si compilasse⁴:

```
\documentclass{minimal}
\makeatletter
\let\@docente\empty
\def\docente#1{\def\@docente{#1}}
\def\getcognome#1_#2!{\def\Cognome{#2}}
\begin{document}
\docente{Mario_Rossi}
\makeatletter
\expandafter\getcognome\@docente!
```

2. Tutti i comandi che terminano un ambiente sono indicati dal compositore con `\end{ambiente}`, ma, dopo un poco di amministrazione e di pulizia eseguito dal comando `\end`, il comando che effettivamente chiude l'ambiente è costituito dal nome stesso dell'ambiente preceduto da `\end` senza l'uso delle graffe. In altre parole, l'ambiente `quote` viene effettivamente chiuso da `\endquote`.

3. Per i nomi e/o i cognomi formati da più parole, bisognerà unirle, lasciando le maiuscole, in modo che la stringa contenga effettivamente un solo spazio: PierPaolo, invece di Pier Paolo; DellaCasa invece di Della Casa. Non è una regola troppo complicata da ricordare.

4. Si ricorda che la classe `minimal` serve appunto per celebrare le definizioni di nuove macro; che `\begin{document}` esegue anche `\makeatother` e che il segno `_` serve per evidenziare gli spazi fra i caratteri.

```
\show\Cognome
\end{document}
```

`\show` mostrerebbe sullo schermo che `\Cognome` è una macro la cui definizione vale proprio `Rossi`.

Il comando `\expandafter` serve per alterare la sequenza delle operazioni: prima viene sviluppata la macro temporanea `\@docente`, che ridiventa la stringa introdotta nel processo di composizione e memorizzata mediante il comando `\docente`, poi questa stringa viene elaborata dal comando `\getcognome` che assegna il solo cognome alla macro `\Cognome`.

3 Il Registro delle Lezioni

Presso il Politecnico di Torino ogni docente deve compilare un Registro delle Lezioni formato da una camicia e tanti intercalari quanti bastano⁵.

La camicia sulla prima pagina, figura 1, riporta le solite intestazioni presenti nei documenti di qualunque università, cioè il nome e il logo; contiene poi il tipo di corsi di studi e la sede di svolgimento dell'insegnamento, il nome dell'insegnamento con il suo codice; se questo insegnamento aggrega anche studenti di altri corsi di studi, vanno elencati tutti i codici e in chiaro la sigla di ciascun corso di laurea; ci vuole evidentemente il nome del docente e la sua facoltà di appartenenza (non necessariamente coincidente con quella della facoltà a cui afferisce l'insegnamento); ci vuole l'anno accademico e il periodo didattico⁶.

La quarta pagina della camicia, figura 2, contiene due tabelle: una serve per indicare quanta didattica riceve ogni studente secondo l'organizzazione del corso, e la seconda indica quanta didattica hanno fornito il docente e i suoi collaboratori: il corso potrebbe essere diviso in squadre e, a seconda della tipologia di attività, potrebbero esserci diversi docenti compresenti; le attività sono divise per tipologie identificate da sigle: L per le lezioni, EA per le esercitazioni in aula, EL per le esercitazioni in laboratorio sperimentale, eccetera.

Gli intercalari, figura 3, contengono una tabella per ogni docente (principale o collaboratore) che forma una 'riga' di una grande tabella che prende tutta la pagina.

Ogni riga deve contenere oltre la data, l'orario, la tipologia di didattica a cui si riferisce quella registrazione, le ore complessive dedicate a quella tipologia di attività, la lingua in cui si svolge

5. La camicia è una cartellina costituita, volendo, da un foglio A3 piegato a metà; ogni intercalare è un foglio A4 inserito ordinatamente dentro la cartellina. In pratica, vista la rarità delle stampanti A3 negli uffici dei docenti, si può stampare tutto su fogli A4, mettere la quarta pagina di copertina come ultima pagina, con la parte scritta all'esterno del fascicolo, e infine pinzare il tutto con punti metallici.

6. In altre università i corsi, forse, sono solo annuali e/o semestrali; presso il Politecnico ci sono corsi annuali, semestrali, trimestrali e bimestrali; si svolgono in periodi successivi, detti periodi didattici.

l'attività, la squadra per la quale l'attività è stata svolta, il numero di docenti compresenti; infine il nome del docente specifico e la sua firma, che ovviamente dovrà essere autografa, quindi la classe del documento deve provvedere solo alla composizione del riquadro destinato alla firma. Ovviamente il riquadro più grande di questa 'riga' contiene il testo che descrive la materia trattata in quella lezione.

Come si vede, il registro richiede una composizione piuttosto complessa; la necessità di ripetere alcune informazioni in righe successive dove cambia solo il nome del docente richiede la formazione di liste; la necessità di eseguire i semplici calcoli richiesti implica altre liste, docente per docente; la necessità di eseguire un minimo di controlli richiede ulteriori comandi.

4 La classe per la composizione dei registri

L'idea di creare una classe per questo tipo di composizione, non è solo quella di comporre bene il registro mediante l'uso dello strumento compositivo migliore, ma anche di sfruttare LATEX per fargli eseguire le scritture ripetitive, nonché per fargli fare i calcoli necessari; le capacità di calcolo di LATEX sono piuttosto limitate, ma, senza nemmeno ricorrere al pacchetto di estensione `calc`, si riesce a fare tutte le somme necessarie per la quarta pagina della copertina sia per righe sia per colonne.

Le liste di docenti e di tipologie di attività didattiche richiedono altre liste. I controlli potrebbero anch'essi avvantaggiarsi dalle macro con argomenti delimitati.

Qui non si descriverà tutta la classe; la descrizione completa può essere ottenuta compilando con LATEX il codice documentato contenuto nel file `registro07.dtx`, BECCARI (2007). La documentazione serve per la corretta manutenzione del codice e non è certamente una lettura divertente. Qui ci si limita solamente a descrivere sommariamente quello che interessa per la spiegazione delle macro con argomenti delimitati.

Si prevede che il docente componga due file; il primo sarebbe il master-file che contiene solo le informazioni per la camicia e per eseguire la lettura del secondo; questo secondo file dovrebbe contenere solo le informazioni per gli intercalari; questa divisione dei ruoli non è essenziale, ma è comoda per molti motivi, specialmente quando sono coinvolti corsi omonimi paralleli⁷.

Per non dover riscrivere le righe 2, 3 e 4 dell'intercalare della figura 3, bisogna che il codice da inserire nel file sorgente sia ridotto al minimo indispensabile; nello stesso tempo il codice non deve essere molto diverso da quello usato per le righe 1 e 5, che si riferiscono ad un solo docente:

7. Questi sono i corsi che dovrebbero procedere in modo sincrono, visto che contengono allievi divisi fra i corsi in base alla lettera iniziale del cognome.

il titolare per la riga 1 e un collaboratore per la riga 5; solo il contenuto della lezione deve variare, evidentemente, ancorché esso debba essere lo stesso per le tre righe 2, 3 e 4.

Il codice della riga 1, relativa al solo titolare il cui nome è già stato memorizzato con i comandi specificati per creare la prima pagina della camicia, dovrebbe essere semplicemente:

```
\begin{lezione}{2004-09-22}%
  {12:30--14:30}{L}{2}
  Introduzione al corso.
```

```
Ripasso ... più comuni.
\end{lezione}
```

Il codice per la riga 5, relativo ad un solo collaboratore dovrebbe essere semplicemente:

```
\begin{lezione}{2004/09/24}%
  {10:30--12:30}{EA}{2}(1,2/2,EN)%
  [GianPaolo Neri]
  Esercizi 1.4, ..., 1.7
\end{lezione}
```

Infine per le tre righe 2, 3 e 4 occorre la lista dei docenti:

```
\begin{lezione}{2004/09/23}%
  {10:30--12:30}{EA}{2}(3,1/2)%
  [Mario Rossi,Fabio Bianchi,%
   GianPaolo Neri]
  Nozioni elementari sui trasformatori.


  Esercizi 1.1, 1.2, 1.3
\end{lezione}
```

Questa descrizione permette di capire che ci vogliono due diversi tipi di argomenti facoltativi: il primo è racchiuso fra parentesi tonde e contiene nell'ordine: (a) il numero di docenti compresenti; (b) il numero della squadra riferito al numero totale di squadre; (c) la lingua in cui si svolge l'attività.

Il secondo tipo di argomento facoltativo, racchiuso fra parentesi quadre, indica una lista di docenti i cui nomi e cognomi sono separati da virgole senza spazi superflui⁸, eventualmente ridotta ad un solo nominativo se il docente corrispondente, diverso dal titolare, è l'unico presente in aula. Non c'è mai bisogno di indicare espressamente il titolare, tranne nei casi in cui egli sia compresente con altri docenti.

Si noti che non è necessario specificare la lingua italiana, perché è quella di default; così non è necessario specificare le squadre se ce ne è una sola; non è nemmeno necessario specificare il numero di docenti compresenti se ce ne è uno solo; quindi l'argomento facoltativo fra parentesi tonde ha, per

8. In generale non è necessario spezzare le righe come fatto sopra, ma se lo si fa, è necessario inserire i segni di commento senza lasciare spazi superflui.



POLITECNICO DI TORINO
FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE

Corsi di Laurea – Sede di Torino

Registro del corso di **ELETTROTECNICA II**
07AUQBT, 07AUQBW,
07AUQCT / IFM+FIS+IFF

Prof. **MARIO ROSSI**
Dipartimento di Elettronica

Anno Accademico **2004/2005**
1° PD

Visto
IL PRESIDE DELLA FACOLTÀ

FIGURA 1: Prima pagina della camicia

“Elettrotecnica II” – Mario Rossi – 07AUQBT, 07AUQBW, 07AUQCT/IFM+FIS+IFF

ORGANIZZAZIONE DEL CORSO

Tipologia di didattica	Numero ore	N. squadre	N. docenti compresenti
L Lezioni	40	1	1
AL Altre lezioni	0	0	0
EA Esercitazioni in aula	4	2	3
EL Esercitazioni in laboratorio	0	0	0
TU Tutorato	0	0	0
VG Visite guidate	0	0	0
LI Laboratori progettuali	0	0	0
Carico didattico totale	44	0	0

NUMERO DI ORE DI ATTIVITÀ PRESTATATA

Docente/coadiutore	Qualifica	Fascia	L	AL	EA	EL	TU	VG	LI	TOT
Mario Rossi	PD	E3	38	2						40
Fabio Bianchi	RC	E3		4						4
GianPaolo Neri	PA	E3		4						4
Totale			38	10						48

IL DOCENTE
.....
(prof. Mario Rossi)

FIGURA 2: Quarta pagina della camicia

Insegnamento: Elettrotecnica II Codici: 07AUQBT, 07AUQBW, 07AUQCT

Data	Tipo	N° Docenti compresenti	Argomento	Docente
2004-09-22	L	1	Introduzione al corso.	Mario Rossi
Orario	N.ore	Squadra di squadre	Ripasso di elettrotecnica; le leggi delle tensioni e delle correnti, i nodi indipendenti, i tagli indipendenti, le maglie indipendenti; le equazioni del connettore. Le leggi costitutive dei componenti più comuni	Firma
12:30–14:30	2	1/1		IT
2004/09/23	EA	3	Nozioni elementari sui trasformatori.	Mario Rossi
Orario	N.ore	Squadra di squadre	Esercizi 1.1, 1.2, 1.3	Firma
10:30–12:30	2	1/2		IT
2004/09/23	EA	3	Nozioni elementari sui trasformatori.	Fabio Bianchi
Orario	N.ore	Squadra di squadre	Esercizi 1.1, 1.2, 1.3	Firma
10:30–12:30	2	1/2		IT
2004/09/23	EA	3	Nozioni elementari sui trasformatori.	GianPaolo Neri
Orario	N.ore	Squadra di squadre	Esercizi 1.1, 1.2, 1.3	Firma
10:30–12:30	2	1/2		IT
2004/09/24	EA	1	Esercizi 1.4, 1.5, 1.6, 1.7	GianPaolo Neri
Orario	N.ore	Squadra di squadre		Firma
10:30–12:30	2	2/2		EN

FIGURA 3: Prime righe di un intercalare

così dire, una importanza minore rispetto a quello racchiuso fra parentesi quadre e all'occorrenza quello può mancare anche se questo è presente.

Similmente i tre parametri dell'argomento facoltativo fra parentesi tonde sono elencati gerarchicamente, nel senso che indicare il primo non richiede di indicare il secondo e il terzo, ma indicare il secondo richiede che si indichi il primo; all'occorrenza, se va bene il valore di default, gli argomenti non indicati possono essere omissi, purché non si omettano le virgole; in altre parole va bene scrivere (, ,EN) ma non va bene indicare solamente (EN): al contrario si può indicare sia (2) sia (2, ,), perché producono lo stesso risultato.

A parte queste specificazioni, ci si rende subito conto che sia il primo, sia il secondo argomento facoltativo sono liste di stringhe separate da virgole; la tipica situazione nella quale per esaminarle bisogna fare uso di macro con argomenti delimitati.

5 Alcune macro con argomenti delimitati

In teoria una macro può avere simultaneamente sia argomenti delimitati sia argomenti non delimitati; per chiarezza di programmazione sarebbe meglio non ridursi a queste situazioni ibride, ma non è vietato. Una macro avente solo argomenti delimitati segue la sintassi seguente⁹:

```
\def<macro><delim0>#1<delim1>#2<delim2>...%
#n<delimn>{<definizione>}
```

con $n \leq 9$. Il nome della *<macro>* può essere qualsiasi, ma per evitare inconvenienti è meglio fare uso del test `\ifdefinable`. Gli argomenti, come è noto, non possono essere più di nove, e corrispondentemente non ci possono essere più di dieci delimitatori. Questi a loro volta possono essere costituiti da qualunque stringa, anche dei semplici spazi, o, eventualmente, delle stringhe che hanno la struttura di comandi veri o non definiti; solo *<delim₀>* non può essere una lettera, perché potrebbe essere confusa con l'ultima lettera della *<macro>*.

Esaminando il codice del pacchetto `babel`, si possono per esempio trovare le definizioni di due comandi usati molto frequentemente in quel codice, introdotti apposta per rendere 'robusti' i comandi condizionali; semplificando un poco, si tratta di:

```
\def\bb1@afterelse#1\else#2\fi{\fi#1}
\def\bb1@afterfi#1\fi{\fi#1}
```

La lettura di questi due comandi permette di capire che servono per 'gettare' la prima clausola di un

9. Qui si fa un esempio che usa `\def` come comando di definizione; si potrebbero usare anche `\gdef`, `\edef` e `\xdef`; `\def` e `\edef` eseguono definizioni locali al gruppo entro il quale vengono eseguite, mentre `\gdef` e `\xdef` eseguono definizioni globali; `\def` e `\gdef` eseguono le definizioni senza sviluppare le macro eventualmente presenti nella *<definizione>*, mentre `\edef` e `\xdef` le sviluppano.

comando condizionale, oppure la seconda clausola oltre la fine dell'espressione condizionale; per esempio, se si volesse controllare se questa pagina sia dispari si potrebbe scrivere in termini di comandi primitivi, estesi con i due comandi di `babel` citati sopra:

```
\expandafter\ifodd\value{page}%
\bb1@afterelse dispari\else
\bb1@afterfi pari\fi
```

I separatori `\else` e `\fi`, benché siano anche dei comandi primitivi, non vengono eseguiti, quando sono letti come separatori, ma servono solo per delimitare la prima clausola dalla seconda; la chiusura dell'istruzione condizionale `\ifodd` viene eseguita dal comando `\fi` contenuto dentro la definizione.

Tornando alle liste formate dai comandi facoltativi della classe dei registri, vediamo dunque che la separazione dei parametri fra parentesi tonde si può eseguire facilmente con i comandi seguenti¹⁰:

```
\def\set@numeri(#1){\lista@numeri#1,,,!}
%
\def\lista@numeri #1,#2,#3,#4!{%
\def@tempA{#1}%
\ifx@tempA\empty\NumDoc=\@ne\relax
\else\NumDoc=#1\relax\fi
%
\def@tempA{#2}%
\ifx@tempA\empty\def\Squadre{1/1}%
\else\def\Squadre{#2}\fi
%
\def@tempA{#3}%
\ifx@tempA\empty\def\Lingua@{\@Lingua}%
\else\def\Lingua@{#3}\fi
}%
```

Il trucco è semplice: `\set@numeri` estrae l'argomento facoltativo delimitato dalla coppia di parentesi tonde (i delimitatori), poi lo passa a `\lista@numeri`, allungandolo con altri tre elementi della lista 'vuoti', poi completa la lista delimitandola con un punto esclamativo.

`\lista@numeri` a sua volta prende la lista, eventualmente allungata, ma la separa in quattro elementi: il primo dovrebbe rappresentare il numero di docenti composti, il secondo il numero relativo di squadre e il terzo la lingua; se il compositore avesse specificato solo il primo, i tre elementi vuoti aggiunti alla lista assicurano che l'insieme di elementi che formano l'argomento facoltativo siano tutti presenti, eventualmente vuoti; se invece il compositore li ha specificati tutti e tre, il quarto argomento raccoglie tutto quel che resta della lista, eventualmente una sequenza di virgole. Le istruzioni successive copiano ciascun valore della lista

10. `\NumDoc` è un contatore interno definito 'alla plain T_EX', con il quale è possibile fare le assegnazioni in modo più agevole, ma non li si può usare 'alla L^AT_EX', per esempio, per i riferimenti incrociati.

in una macro temporanea `\@tempA`; questa viene confrontata con la macro vuota `\empty`; se entrambe sono vuote, allora assegna il valore di default, altrimenti assegna il parametro specificato.

La lista dei docenti è più complicata da separare nei suoi elementi, perché i nominativi dei docenti sono in numero imprecisato e, memoria dell'elaboratore permettendo, in numero arbitrariamente grande. Quindi bisogna estrarne uno alla volta e ciclicamente verificare che cosa resta nella lista. Ma per ciascun nominativo estratto bisogna anche comporre la riga nel registro e raccogliere i dati numerici di impegno per ogni tipo di attività eseguito dal docente che si sta elaborando.

Un primo comando carica la lista dei docenti estraendola dall'argomento fra parentesi quadre:

```
\def\set@docente[#1]{%
    \edef\lista@docenti{#1}%
% seguono altri comandi inessenziali per
% questa esposizione; infine si esegue:
\componi@scatole}
```

Successivamente la `\lista@docenti` viene usata dal comando ciclico che ripete la composizione della riga dell'intercalare per ogni docente elencato:

```
\def\componi@scatole{%
\@ripeti@@@docente:=\lista@docenti\do{%
\expandafter\getcognome\@@@docente!%
\expandafter\ifx
    \cname\Cognome\endcname\relax%
\expandafter\gdef
    \cname\Cognome\endcname{EA=0}%
\fi
\start@boxes\lez@i\lez@ii\lez@iii%
    \lez@iv\lez@v\lez@vi\lez@vii
\emettiscatola
}}

\def\@ripeti#1:=#2\do#3{%
    \expandafter\def\expandafter%
        \@tempA\expandafter{#2}%
    \ifx\@tempA\@empty \else
    \expandafter\@riciclo#2,,\@#1{#3}\fi
}%

\def\@riciclo#1,#2,#3\@@#4#5{\def#4{#1}%
#5%
\def#4{#2}%
\ifx #4\@empty
\expandafter\@finisci%
\else
    \expandafter\@riciclo\fi
#2,#3,\@nil\@@#4{#5}%
}

\def\@finisci#1\@@#2#3{%
```

Il comando `\componi@scatole` esamina la lista e per ogni docente compreso nella lista definisce `\@@@docente`; si noti il costrutto:

```
\@ripeti<macro>:=<lista>\do{<comandi>}
```

dove `<macro>` e `<lista>` sono argomenti delimitati.

Questo è un tipico costrutto informatico per ripetere una serie di `<comandi>` finché una certa variabile segue una certa enumerazione rappresentata da numeri, da lettere, o da elementi ordinati in una lista. Il nucleo di L^AT_EX contiene una simile macro `\@for` la cui sintassi è identica a quella di `\@ripeti`, ma il cui svolgimento è diverso, perché è prevista per un uso più generale. Nello stesso tempo i delimitatori `:=` e `\do`, al di là della loro funzione strettamente meccanica di delimitatori, esprimono concetti che al compositore sono più chiari di qualunque altro segno scelto a caso, anche se potrebbe svolgere la stessa funzione delimitatrice.

`\@ripeti` inizialmente esamina la lista dopo averla copiata in `\@tempA`; se questa è vuota, non fa nulla, ma se contiene qualcosa allora procede e passa i suoi argomenti al comando ad argomenti delimitati `\@riciclo`; si vede che la lista su cui operare è prolungata con altri due elementi vuoti e il delimitatore finale è costituito dal delimitatore `\@@` prima di indicare gli ultimi due argomenti non delimitati.

`\@riciclo` prende cinque argomenti, i primi tre dei quali sono delimitati da virgole e terminati con `\@@`; ma si noti: i primi tre argomenti contengono rispettivamente il primo nominativo, il secondo nominativo (eventualmente vuoto) e poi la lista residua dei nominativi dopo i primi due (la lista residua potrebbe essere costituita da una lista vuota, contenente solo le virgole).

Viene assegnato il primo nominativo alla `<macro>` che nel nostro caso è `\@@@docente`, poi si esegue la lista di comandi contenuta nel quinto argomento. Fatto questo la macro deve decidere se proseguire o terminare.

Per questo scopo assegna il secondo elemento della lista al quarto argomento, cioè a `\@@@docente`; se questo è vuoto non bisogna continuare, altrimenti bisogna cominciare un nuovo ciclo; nello stesso tempo bisogna anche usare gli argomenti residui.

Se dunque il prossimo nominativo è vuoto, viene chiamato il comando `\@finisci` che si mangia tutta la lista rimanente e tutti gli argomenti rimanenti senza fare assolutamente nulla d'altro; altrimenti viene chiamato di nuovo il comando `\@riciclo` accodando alla lista un oggetto qualsiasi, che non verrà mai esaminato, perché nella lista è preceduto da almeno un nominativo vuoto.

Un altro uso particolare delle macro con argomenti delimitati è quello con il quale viene costruita una lista di informazioni nella macro identica al cognome del docente. Si riprendono alcuni comandi riportati sopra, ma sui quali si è evitato di fare commenti:

```
\expandafter\ifx
    \cname\Cognome\endcname\relax%
```

```

\expandafter\gdef
\csname\Cognome\endcsname{EA=0}%
\fi

```

Questa serie di brevi comandi serve per definire una macro col nome identico al cognome del docente: se il nominativo del docente era Mario Rossi, il comando `\Cognome`, estratto con `\get@cognome`, contiene Rossi come sua definizione.

Ricordiamo che i comandi `\csname` e `\endcsname` racchiudono una stringa con la quale costruiscono il nome di una macro; l'`\expandafter` messo prima del test `\ifx` serve per sviluppare `\csname` prima di eseguire il test; `\csname` esamina i token successivi fino al comando `\endcsname` e, se ce ne è qualcuno sviluppabile, lo sviluppa; nel nostro esempio sviluppa `\Cognome` e quindi la coppia `\csname` e `\endcsname` crea il nome della macro `\Rossi` con la particolarità che le assegna di default l'equivalenza con `\relax` nel caso che questo nome di macro non abbia già una definizione: la stessa cosa succede con il comando successivo. Tutto il costruito nel nostro esempio diventa quindi equivalente a:

```
\ifx\Rossi\relax \gdef\Rossi{EA=0}\fi
```

Tutta quella lunga sequela di comandi diventa un semplice trucco per procedere nella stessa maniera qualunque sia il cognome del docente.

In relazione alle righe 2, 3 e 4 della figura 3, dopo l'esecuzione dei comandi ciclici eseguiti nello sviluppo della lista di nominativi, esisteranno le macro `\Bianchi` che vale `EA=0,EA=2`; `\Rossi` che vale `LL=2,EA=2` grazie al fatto che prima di quella esercitazione in aula il docente aveva già fatto una lezione di due ore, e `\Neri` che vale `EA=0,EA=2`.

Tutte queste liste verranno poi esaminate come argomenti delimitati da altre macro; i delimitatori ora sono alternativamente un segno di = e una virgola; quest'ultima serve per isolare un elemento alla volta, mentre il primo serve per separare il tipo di attività didattica dalle ore svolte. Nuovamente due macro ad argomenti delimitati serviranno per estrarre gli elementi della lista e, all'interno di queste, il codice dell'attività e il tempo corrispondente. Esse serviranno per comporre l'ultima tabella della figura 2 per inserire i numeri giusti nelle caselle corrette e per fare le somme per colonne e per righe.

6 Conclusione

Nel TEXbook, (KNUTH, 1990, capitolo 20 e appendice D), si descrivono le macro con le definizioni

eseguite mediante i comandi primitivi; vi si trovano anche molti esempi di uso di simili macro. Il file `latex.ltx` (che si trova nella distribuzione del sistema TEX all'indirizzo `.../texmf-dist/tex/latex/base/`) ne contiene moltissime altre; ovviamente si possono ristudiare le macro appena descritte nel file `registro07.dtx`, BECCARI (2007) o meglio, nella sua composizione mediante LATEX.

Quello che si è esposto qui non serve per spaventare i possibili utenti, ma per indirizzare gli scrittori di file di classe o di estensione verso comandi che rendono il codice più leggibile e quindi più facile da correggere o da modificare.

Si consiglia di usare delimitatori intelligibili; non sempre è necessario, ma più complicato è un comando, tanto più espliciti dovrebbero essere i nomi dei comandi e quelli dei delimitatori; così facendo, il codice sarà immediatamente comprensibile senza bisogno di complicati commenti o frasi circonvolute.

Certamente si sconsiglia vivamente di usare questi comandi nel corpo di un file sorgente di un documento da comporre; se li si usa, è bene farlo in file di classe o di estensione. D'altra parte Leslie Lamport, il padre di LATEX, è stato piuttosto esplicito: egli ha definito e usato moltissimi comandi con argomenti delimitati nel nucleo di LATEX, ma non ne ha lasciato praticamente nessuno a disposizione dei compositori; questi li possono usare (senza saperlo: per esempio quando usano il comando `\cline` per comporre tabelle; quale delimitatore viene usato?) e non ha creato nessun comando di alto livello per definirli. Questo è un segno evidente che si tratta di cose delicate da non lasciare in mano al compositore, il quale è invece invitato a concentrarsi sul contenuto, piuttosto che su come comporlo.

Riferimenti bibliografici

- BECCARI, C. (2007). «Il pacchetto `registro07`». `registro07.dtx`. URL `ftp://ftp.polito.it/pub/tex/polito/registro/registro07.zip`.
- KNUTH, D. E. (1990). *The TEXbook*, volume A di *Computers and Typesetting*. Addison Wesley, 18^a edizione.

▷ Claudio Beccari
claudio dot beccari at
alice.it