Introduzione al pacchetto *xparse*

Claudio Beccari

GUIT *meeting* 2018

Introduzione

Lo scopo di questa presentazione è quello di mostrare alcuni esempi d'uso del pacchetto *xparse* con il quale si possono definire nuovi comandi e nuovi ambienti in modo molto più efficace che con i comandi nativi di LaTeX.

Tutti conoscono i comandi nativi di LaTeX.

I comandi nativi di LaTeX

```
\newcommand{⟨macro⟩}[⟨N.arg⟩]{⟨definizione⟩}
\renewcommand{⟨macro⟩}[⟨N.arg⟩]{⟨definizione⟩}
\providecommand{⟨macro⟩}[⟨N.arg⟩]{⟨definizione⟩}
\DeclareRobustCommand{⟨macro⟩}[⟨N.arg⟩]%
    {⟨definizione⟩}

\newenvironment{⟨ambiente⟩}[⟨N.arg⟩]%
    {⟨apertura⟩}{⟨chiusura⟩}
\renewenvironment{⟨ambiente⟩}[⟨N.arg⟩]%
    {⟨apertura⟩}{⟨chiusura⟩}
```

I comandi nativi di LaTeX: pregi

Non c'è dubbio che il pregio maggiore è la semplicità di queste definizioni.

Siamo talmente abituati ad usarli che non ci accorgiamo delle limitazioni.

Certo per definizioni semplici non occorre ricorrere ai grandi mezzi di *xparse*.

Il comandi nativi di LaTeX: difetti

Separiamo i difetti di **\newcommand** e soci da quelli di **\newenvironment** e del suo unico socio.

Difetti di \newcommand

Il comando definito può essere fragile; consente l'uso di un solo argomento facoltativo, ma delimitato solo da parentesi quadre; l'argomento facoltativo può essere solo il primo; è un po' strana la sintassi per specificarne un eventuale valore predefinito; mancano comandi per testare la presenza di un valore predefinito.

Assegnare un valore predefinito

\newcommand{⟨*macro*⟩}[1][⟨*val.pred.*⟩]{⟨*definizione*⟩}

In altre parole occorre un secondo argomento facoltativo la cui presenza garantisce che il primo argomento sia facoltativo; la sua assenza indica che il primo argomento è obbligatorio.

Difetti di \newenviroment

Innanzi tutto questo comando manca del “socio” **\provideenvironment**.
Inoltre solo il comando di apertura può disporre di argomenti che hanno gli stessi inconvenienti segnalati per **\newcommand**; il comando di chiusura non può fare uso degli argomenti specificati per il comando di apertura che vanno quindi memorizzati in opportune macro da usare in chiusura.

Come rimediare?

Ci sono essenzialmente poche strade per superare gli inconvenienti segnalati.

- Ricorrere ai comandi nativi di TEX. La strada è senz'altro percorribile, ma richiede abilità di programmazione non indifferenti.
- Ricorrere alle opzioni del tipo chiave=valore. Anche questa strada è percorribile, ma la documentazione dei pacchetti che consentono di usare questa tecnica richiede la mentalità del programmatore per essere capita.
- Ricorrere a *xparse*. Questa è la strada che percorreremo qui.

I comandi di definizione di *xparse*

xparse mette a disposizione diversi comandi per definire le sue funzioni, che non si chiamano più macro per vari motivi che qui si tralasciano.

Comandi per definire funzioni

\NewDocumentCommand{⟨*descrittori*⟩}{⟨*definizione*⟩}
\RenewDocumentCommand{⟨*descrittori*⟩}{⟨*definizione*⟩}
\ProvideDocumentCommand{⟨*descrittori*⟩}{⟨*definizione*⟩}
\DeclareDocumentCommand{⟨*descrittori*⟩}{⟨*definizione*⟩}

La ⟨*definizione*⟩ viene fatta nel solito modo, ma la novità sono i ⟨*descrittori*⟩ degli argomenti

I comandi di definizione di *xparse*

xparse mette a disposizione anche comandi per definire le necessarie funzioni per gli ambienti.

Comandi per definire ambienti

```
\NewDocumentEnvironment{⟨ambiente⟩}{⟨descrittori⟩}%
    {⟨apertura⟩}{⟨chiusura⟩}
\RenewDocumentEnvironment{⟨ambiente⟩}{⟨descrittori⟩}%
    {⟨apertura⟩}{⟨chiusura⟩}
\ProvideDocumentEnvironment{⟨ambiente⟩}{⟨descrittori⟩}%
    {⟨apertura⟩}{⟨chiusura⟩}
```

Definizioni sviluppabili

xparse mette a disposizione altrettanti comandi per definire funzioni sviluppabili (o espandibili). Questo è un argomento per programmatori avanzati e qui non ne parlo.

I descrittori degli argomenti

Come si vede *xparse* non richiede più solo il numero di argomenti ma richiede per ogni argomento un descrittore (al massimo nove argomenti).

Tutte le funzioni definite con i suoi comandi sono robuste.

Dispone di molti descrittori per argomenti obbligatori, per quelli facoltativi, per quelli con funzione booleana, per quelli delimitati, per quelli a cui assegna un valore predefinito; mette a disposizione i test necessari per verificare la semplice presenza degli argomenti booleani, o per verificare se a un dato argomento facoltativo privo di default è stato assegnato un valore.

Descrittori per argomenti obbligatori

Ecco i descrittori degli argomenti.

Argomenti obbligatori

`m` descrive un argomento obbligatorio da racchiudere fra graffe.

`r`⟨*car1*⟩⟨*car2*⟩ descrive un argomento obbligatorio da racchiudere fra ⟨*car1*⟩ a sinistra e ⟨*car2*⟩ a destra.

`v` descrive un argomento obbligatorio da leggere verbatim; può essere racchiuso fra graffe, ma può essere anche delimitato fra due caratteri identici, come fa il comando `\verb`.

Descrittori per argomenti booleani

Gli argomenti booleani sono singoli token di cui viene rilevata solo la presenza, un po' come si fa con l'asterisco dei comandi nativi di LaTeX.

Descrittori per gli argomenti booleani

`s` serve per dichiarare che questo è un segnaposto per l'asterisco; la sua presenza può essere controllata per decidere se una funzione debba eseguire operazioni diverse a seconda che l'asterisco sia o non sia presente.

`t`⟨*car*⟩ serve per dichiarare il token ⟨*car*⟩ come argomento booleano. Anche questo segnaposto funziona come quello per l'asterisco.

Descrittori per gli argomenti facoltativi

Gli argomenti facoltativi possono essere variamente delimitati con descrittori appositi i quali possono anche specificare un valore predefinito.

Descrittori per gli argomenti facoltativi

o e O{⟨*default*⟩} descrive un segnaposto per un argomento facoltativo da racchiudere fra parentesi quadre; il descrittore minuscolo non accetta argomenti; quello maiuscolo accetta come argomento il valore di ⟨*default*⟩.

d⟨*car1*⟩⟨*car2*⟩ e D⟨*car1*⟩⟨*car2*⟩{⟨*default*⟩} svolgono funzioni analoghe ai due descrittori precedenti, salvo che l'argomento facoltativo deve essere delimitato a sinistra da ⟨*car1*⟩ e a destra da ⟨*car2*⟩.

Altri descrittori

Il pacchetto *xparse* definisce anche altri descrittori, alcuni dei quali sono ancora allo stato sperimentale. Quindi non ne dico nulla.
Quelli che ho descritto sono già abbastanza per permettere di creare macro potentissime con una sola definizione di solito relativamente semplice.
Certo: "di solito" significa che invece può esserci la necessità di creare funzioni che facciano molte cose e le facciano sotto il controllo di molti argomenti facoltativi.
Qui mostrerò alcune applicazioni per rendere l'idea di che cosa si può fare.

Funzioni di controllo

È però necessario descrivere le funzioni per controllare gli argomenti booleani e quelli facoltativi.

Controllo booleano

```
\IfBooleanTF{⟨numero argomento⟩}{⟨vero⟩}{⟨falso⟩}
\IfBooleanT{⟨numero argomento⟩}{⟨vero⟩}
\IfBooleanF{⟨numero argomento⟩}{⟨falso⟩}
```

`T` e `F` indicano le due alternative del test; se l'argomento booleano è presente vale `T`, altrimenti vale `F`; ⟨*vero*⟩ è quello che c'è da fare se il test è vero, e ⟨*falso*⟩ se il test è falso; come si vede le ultime due righe sono scorciatoie per dire di non fare niente in uno dei due casi.

Il ⟨*numero argomento*⟩ indica come al solito il numero d'ordine di un argomento preceduto da #.

Funzioni di controllo

Analogamente esistono due test simili e con le stesse scorciatoie per verificare se un dato argomento facoltativo privo di un valore di default ha ricevuto un valore al momento dell'esecuzione della funzione.

Controllo di valore

\IfValueTF{⟨*numero argomento*⟩}{⟨*vero*⟩}{⟨*falso*⟩}
\IfValueT{⟨*numero argomento*⟩}{⟨*vero*⟩}
\IfValueF{⟨*numero argomento*⟩}{⟨*falso*⟩}

\IfNoValueTF{⟨*numero argomento*⟩}{⟨*vero*⟩}{⟨*falso*⟩}
\IfNoValueT{⟨*numero argomento*⟩}{⟨*vero*⟩}
\IfNoValueF{⟨*numero argomento*⟩}{⟨*falso*⟩}

Il fatto che ci siano due test opposti serve per rendere più chiara la ⟨*definizione*⟩ della funzione.

Esempio: la scacchiera/cruciverba

Nel fascicolo che contiene gli atti di questo meeting si parla di un *Concorso della scacchiera*. Vale la pena di esaminare la soluzione che ho escogitato ricorrendo a *xparse*. Con una sola macro per l'utente ho definito una serie di descrittori obbligatori e facoltativi e di ulteriori possibilità offerte all'utente, tanto che il comando per l'utente può essere usato in 128 modi diversi.

Esempio: la scacchiera/cruciverba

Ma a che cosa servono 128 modi diversi? Non c'è da preoccuparsi; ci sono sette cose facoltative; e a seconda che vengano o non vengano sfruttate si hanno $2^7 = 128$ possibilità diverse. Molte in realtà non danno risultati differenti, ma una cosa è il calcolo delle possibilità e una cosa è quello che si fa in pratica; tuttavia si devono scegliere le dimensioni della scacchiera da destinare ad un gioco o a una decorazione o a un bandiera da Gran Premio, oppure ad un cruciverba; per gli scacchi colorati bisogna scegliere con quali colori; eccetera. Il tutto fatto con un'unica macro più qualche cosetta di servizio.

Esempio: un indice analitico particolare

Scrivendo di LaTeX si desidera caratterizzare con font diversi i vari oggetti che formano la sua sintassi; dai pacchetti, alle classi; dai programmi, ai comandi; e molto altro; per ognuno di questi oggetti si definisce il modo di scriverlo; talvolta lo si vuole anche inviare all'indice analitico. Ricordando che l'asterisco nei comandi di sezionamento serve fra l'altro per non scrivere nulla nell'indice generale, sfruttiamo lo stesso concetto per definire i comandi di scrittura/indicizzazione.

Esempio: un indice analitico particolare

Per esempio vogliamo scrivere una control sequence con un certo stile ma senza eseguirla, con la possibilità di inviarla all'indice analitico. Ecco che *xparse* permette di farlo con una sola macro (e una di servizio);

Indicizzare il nome di una control sequence

```
\DeclareRobustCommand*\csstyle[1]{%
\textnormal{\texttt{\char92#1}%
}}

\NewDocumentCommand{\cs}{s m}{%
\csstyle{#2}\IfBooleanTF{#1}{}{%
\index{#2@\csstyle{#2}}}}
```

Esempio: il frontespizio di *TOPtesi*

L'ambiente per gestire i vari frontespizi che si possono comporre con il pacchetto *TOPtesi* richiedeva una gestione diversa a seconda delle prescrizioni di varie università, ma anche una gestione diversa del logo dell'ateneo al fine di non ripetere due volte quello che già appariva nel logo.
Molti loghi oggi giorno non contengono solo il disegno, di solito di forma circolare, con le insegne dell'ateneo (il timbro), ma contengono anche il suo nome ufficiale (il logotipo) scritto accanto. Se un logo di questo genere viene posto in testa al frontespizio e poi subito sotto si scrive "Università degli studi di. . .", si scrive due volte lo stesso nome a distanza di pochi centimetri; conviene allontanare le due informazioni, oppure omettere la dicitura letterale subito sotto il logotipo.

Esempio: il frontespizio di *TOPtesi*

Per allontanare il logo dall'intestazione basta metterlo nella parte inferiore del frontespizio. A questo scopo l'ambiente di composizione richiede un solo descrittore, il solito asterisco.

Ambiente con asterisco

```
\NewDocumentEnvironment{ThesisTitlePage}{s}
{% APERTURA
...
}{% CHUSURA
\IfBooleanTF{#1}{...}{...}
...
}
```

Si vede che l'argomento "asterisco" ricevuto dal comando di apertura può essere usato nel comando di chiusura.

Esempio: il frontespizio di *TOPtesi*

La cosa nuova nell'uso dell'asterisco nei comandi di apertura e chiusura degli ambienti fa si che l'asterisco non sia più parte del nome dell'ambiente (come per gli ambienti asteriscati nativi di LaTeX), ma è un argomento. Per l'esempio qui proposto l'uso del comando asteriscato diventa il seguente.

Ambiente asteriscato

```
\begin{ThesisTitlePage}*
...
\end{ThesisTitlePage}
```

Questo vale per ogni ambiente definito con i comandi di *xparse*, non solo per quello dell'esempio.

Esempio: Ellissi varie

Per disegnare un'ellisse col centro nell'origine degli assi della tela del disegno e di cui siano specificati solo le lunghezze dei due semiassi non è un problema e nell'articolo si mostra la definizione del comando **\ellisse**; esso ha la sintassi seguente, non identica ma simile alla sintassi di **\circle** per l'ambiente nativo di LaTeX.

Sintassi di \ellisse

\ellisse{⟨*semissasse oriz.*⟩**}{**⟨*semiasse vert.*⟩**}**

Il centro di tale ellisse può essere messo in posizione con il comando **\put**.

Esempio: Ellissi varie

Il codice per disegnare l'ellisse elementare è i seguente.

Ellisse elementare

```
\newcommand\ellisse[2]{%
\bgroup\def\a{#1}\def\b{#2}%
% Calcolo dei punti di controllo
\dimendef\x=256 \x=0.552285\p@
\edef\ax{\strip@pt\dimexpr\a\x\relax}
\edef\bx{\strip@pt\dimexpr\b\x\relax}
% Disegno dell'ellisse
\moveto(\a,0)
\curveto(\a,\bx)(\ax,\b)(0,\b)
\curveto(-\ax,\b)(-\a,\bx)(-\a,0)
\curveto(-\a,-\bx)(-\ax,-\b)(0,-\b)
\curveto(\ax,-\b)(\a,-\bx)(\a,0)
\fillstroke
\egroup}
```

Esempio: Ellissi varie

La particolarità del suo codice è che non termina con uno dei due comandi nativi dell'ambiente ***picture***, esteso con il pacchetto *pict2e*:

- **\strokepath** per disegnare il contorno nero o colorato, oppure
- **\fillpath** per riempire il perimetro di nero o di colore.

ma termina con **\fillstroke** che viene reso equivalente all'uno o all'altro dei comandi suddetti dalla funzione che mette in posizione l'ellisse.

Il colore è quello in vigore nel momento in cui si usa il comando **\ellisse**.

Esempio: Ellissi varie

Vogliamo perciò costruire con *xparse* una funzione che accetti diversi argomenti.

1. Un asterisco facoltativo che funzioni come per `\circle`; se c'è, il contorno è riempito di colore.
2. Facoltativamente le coordinate del centro fra parentesi tonde.
3. Facoltativamente un angolo di rotazione fra parentesi quadre.
4. I due semiassi, evidentemente obbligatori.
5. Facoltativamente il colore di riempimento fra parentesi quadre.
6. Facoltativamente le caratteristiche del contorno fra parentesi quadre.

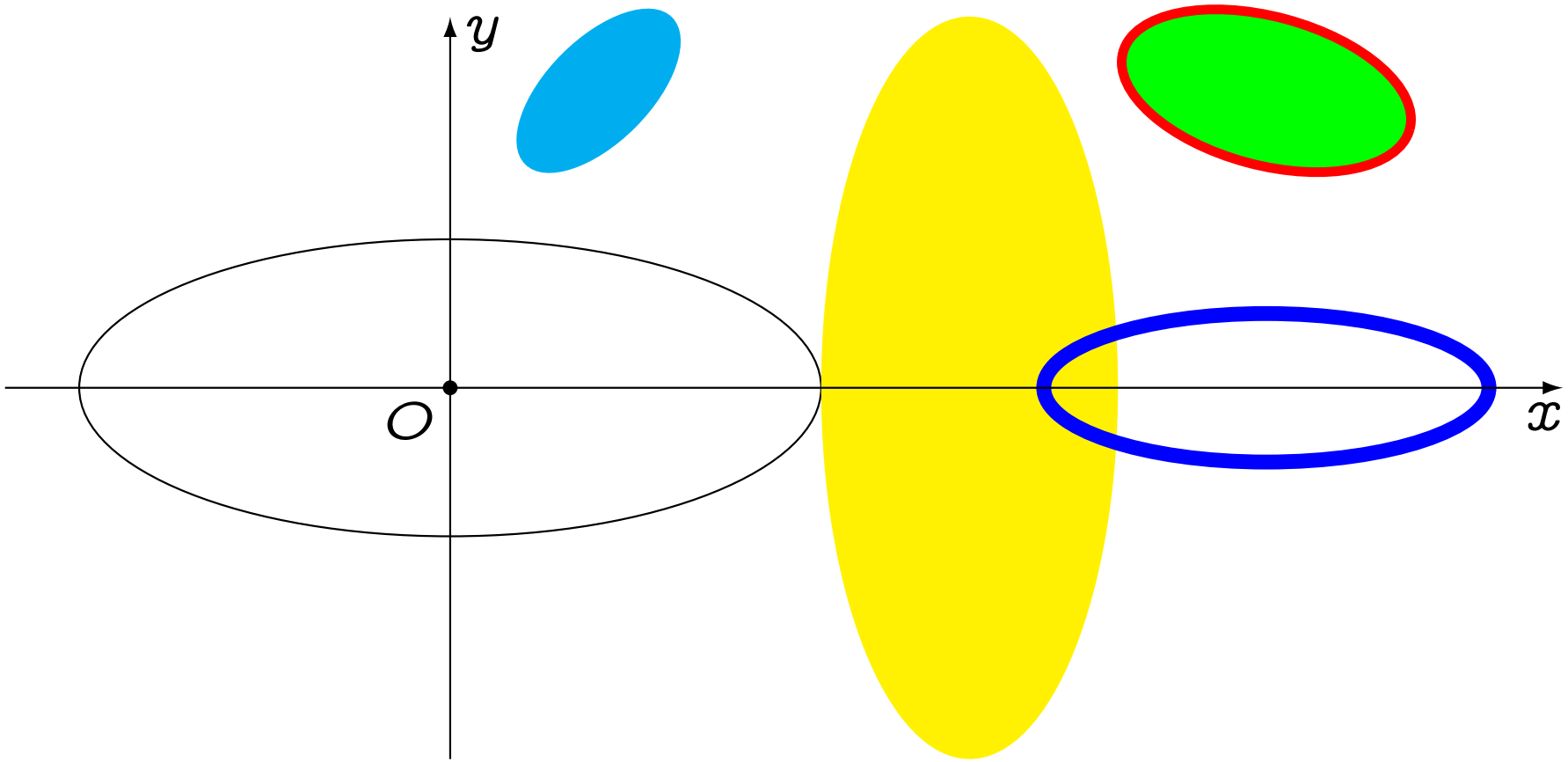
Esempio: Ellissi varie

La definizione della funzione con *xparse* richiede quindi sette descrittori:

Definizione della funzione

```
\NewDocumentCommand{\Xellisse}%
  { s D(){0,0} O{0} m m O{} o }%
    {\IfBooleanTF{#1}%
     {\let\fillstroke\fillpath}%
     {\let\fillstroke\strokepath}%
    \put(#2){\rotatebox{#3}%
      {{#6\ellisse{#4}{#5}}}%
    \IfValueTF{#7}{\let\fillstroke\strokepath
    #7\ellisse{#4}{#5}}{}}}%
    }
```

Esempio: Ellissi varie



Esempio: Ridefinizione di \chapter

la sintassi del comando originale **\chapter** di LaTeX, come sanno tutti gli utenti, è la seguente.

Il comando \chapter originale

\chapter[⟨*titolo breve*⟩]{⟨*titolo lungo*⟩}

oppure

\chapter*{⟨*titolo*⟩}

Come è noto il comando asteriscato non mette nulla nell'indice e non numera il capitolo. Il comando non asteriscato numera il capitolo solo nella main matter usando il ⟨*titolo lungo*⟩ e ne mette il ⟨*titolo breve*⟩ nell'indice e nelle testatine; se il l'argomento facoltativo non è specificato viene reso uguale al ⟨*titolo lungo*⟩.

Esempio: Ridefinizione di \chapter

Il comportamento abbastanza rigido del comando nativo di LaTeX (classe *book*) non permette di inviare alle testatine un titolo diverso sia da quello lungo sia da quello breve.

La classe *memoir* ridefinisce i comando perché si possa usare la sintassi con due argomenti facoltativi:

\chapter di *memoir*

\chapter[⟨*indice*⟩][⟨*testatina*⟩]{⟨*titolo lungo*⟩}

Il difetto è che questa definizione non modifica il comando asteriscato e non permette di specificare ⟨*testatina*⟩ se non si specifica prima ⟨*indice*⟩.

Esempio: Ridefinizione di \chapter

Va notato che le definizioni originali sono formate da catene di macro che leggono un argomento alla volta e ne passano i valore (mediante macro interne) alla macro successiva.

Il meccanismo funziona bene ma ha i limiti che abbiamo indicato sopra.

I programmatori che l'hanno sviluppato hanno impiegato artifici di vario genere che rendono il codice relativamente difficile da leggere per un utente neofita.

Metterci le mani dentro, quindi è altamente sconsigliabile, tanto più che con *xparse* si può fare meglio e prima.

Esempio: Ridefinizione di \chapter

Vogliamo ridefinire il comando **\chapter** in modo da poter specificare ⟨*indice*⟩ e ⟨*testatina*⟩ e che l'argomento ⟨*testatina*⟩ sia usabile anche con il comando asteriscato.

Ricorrendo a *xparse* dobbiamo usare il suo comando **\RenewDocumentCommand** e specificare gli argomenti facoltativi in modo che siano racchiusi fra delimitatori diversi; scegliamo i segni < > come delimitatori.

Esempio: Ridefinizione di \chapter

La ridefinizione del comando è riportata nel suo insieme nell'articolo; qui è troppo lungo e del tutto inutile scendere nei dettagli. Mi limito pertanto a mostrare solo la lista dei descrittori.

Ridefinizione di \chapter

```
\RenewDocumentCommand{\chapter}%
{ s O{??} D<>{??} m }%
{⟨definizione⟩}
```

Come si vede oltre al solito asterisco booleano abbiamo due argomenti facoltativi, il primo delimitato da parentesi quadre e il secondo dai segni < >. Entrambi hanno il valore predefinito ??.

Esempio: Ridefinizione di \chapter

Con tre argomenti facoltativi abbiamo in totale otto possibili comandi;

Le varianti del comando \chapter

\chapter{⟨*titolo*⟩}
\chapter*{⟨*titolo*⟩}
\chapter[⟨*indice*⟩]{⟨*titolo*⟩}
\chapter*[⟨*indice*⟩]{⟨*titolo*⟩}
\chapter<⟨*testatina*⟩>{⟨*titolo*⟩}
\chapter*<⟨*testatina*⟩>{⟨*titolo*⟩}
\chapter[⟨*indice*⟩]<⟨*testatina*⟩>{⟨*titolo*⟩}
\chapter*[⟨*indice*⟩]<⟨*testatina*⟩>{⟨*titolo*⟩}

Esempio: Ridefinizione di \chapter

Nei comandi asteriscati nulla viene mandato all'indice e, se si è specificato l'argomento facoltativo ⟨*indice*⟩, questo viene ignorato.

Quando non vengono specificati gli argomenti facoltativi ⟨*indice*⟩ e/o ⟨*testatina*⟩ l'uno o l'altro o entrambi vengono resi identici a ⟨*titolo*⟩.

Nelle definizioni dei loro valori di default compaiono come valori predefiniti due punti interrogativi; in questo modo è possibile diagnosticare se un dato argomento facoltativo abbia ricevuto un valore diverso da un *blank* (cioè uno spazio), o da nulla del tutto (le parentesi che non delimitano niente, nemmeno uno spazio), oppure una stringa vera e propria.

Esempio: Ridefinizione di \chapter

Nella classe *book* il comando **\tableofcontents** compila l'indice generale con il titolo creato con **\chaper*** e con le testatine uguali ottenute usando **\markboth**. Con quella classe conviene ridefinire **\tableofcontents** così:

Ridefinizione di \tableofcontents

```
\renewcommand\tableofcontents{%
  \if@twocolumn
    \@restonecoltrue\onecolumn
  \else
    \@restonecolfalse
  \fi
  \chapter*{\contentsname}%
  \@starttoc{toc}%
  \if@restonecol\twocolumn\fi
  }
```

Applicazione della funzione \chapter

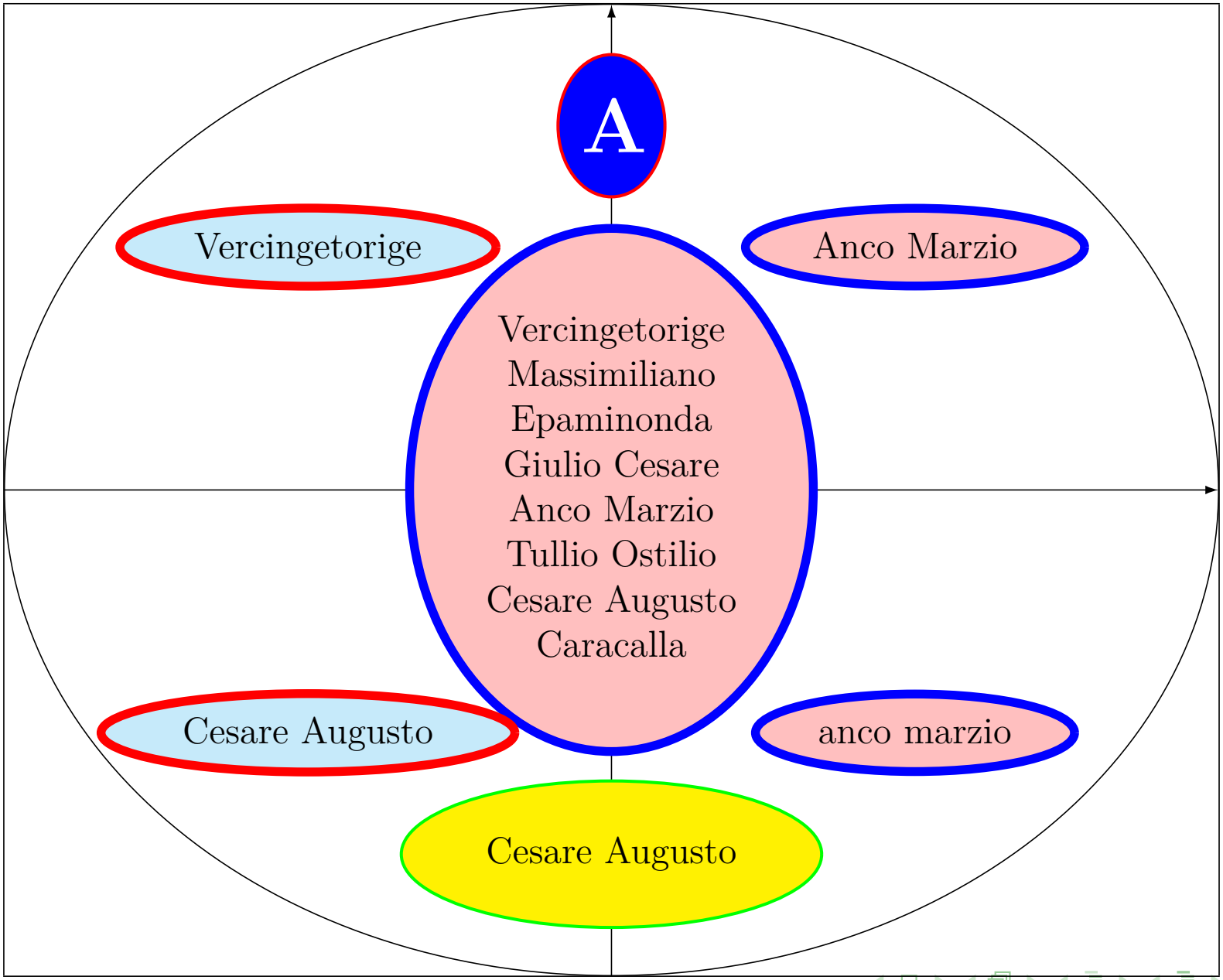
Nell'articolo sono mostrate alcune pagine di un documento di prova composto con la classe *book* dove sono mostrati gli effetti che si ottengono sia nella *front matter*, sia nella *main matter* sia nella *back matter* dei vari modi di usare **\chapter** ridefinito come descritto prima.
Non li mostro qui perché con i quadri di questa presentazione non si vedrebbe granché.

Osservazione 1: si potrebbe seguire una strada analoga per ridefinire anche il comando **\section**.
Osservazione 2: con altre classi le ridefinizioni andrebbero probabilmente eseguite in un altro modo.
Ma l'opportunità di ridefinire i comandi di sezionamento va esaminata con cura; non è da fare solo perché lo si può fare.

Una legenda ellittica

Non ne ho parlato nell'articolo, ma usando *xparse* mi sono divertito a creare una macro con pochi argomenti obbligatori e molti argomenti facoltativi diversamente delimitati per creare legende incorniciare da un ellisse che si adatta da solo al contenuto del testo. Ovviamente posizione della legenda, i colori dello sfondo e dell'eventuale cornice, del suo spessore, del corpo, serie, forma e colore del testo da incorniciare, del margine fra legenda e cornice, sono tutti parametri facoltativi.

Una legenda ellittica



Conclusioni

Credo che gli esempi siano sufficienti a dimostrare la potenza del pacchetto *xparse*.

Certo non vale la pena ricorrere a *xparse* per definizioni semplicissime con applicazioni semplicissime. Non conviene nemmeno per macro non banali, ma tutto sommato semplici come quella dell'ellisse di base **`\ellisse`**.

Ma credo che la potenza e la flessibilità delle funzioni/macro che si possono definire dimostrino ampiamente quanto sia opportuno ricorrere a *xparse* più spesso di quanto non si faccia di solito.

Buon lavoro con LaTeX e con *xparse*