

L'ARTE DI SCRIVERE CODICI CON L^AT_EX

LORENZO PANTIERI

4 dicembre 2008

INDICE

1	Introduzione	1
2	Impostazioni del pacchetto listings	5
3	Personalizzare l'aspetto dei codici	7
3.1	Numerare le righe	7
3.2	Rientri	9
3.3	Riquadri	9
3.4	Sfondi colorati	10
3.5	Evidenziare parole	10
3.6	Inserire una parola nell'indice analitico	11
3.7	L'allineamento delle colonne	11
4	Tecniche avanzate	12
4.1	Ambienti personalizzati	12
4.2	Inserire comandi L ^A T _E X all'interno di un codice	13
4.3	Definire nuovi linguaggi	13
	Riferimenti bibliografici	14

ELENCO DEI CODICI

1	Un esempio di codice mobile	3
2	Un codice "fisso"	4

In questo articolo viene presentato il pacchetto listings, che permette di inserire codici all'interno di un documento. Il pacchetto permette di esercitare un controllo molto preciso sul formato del codice e riconosce un elevato numero di linguaggi di programmazione.

1 INTRODUZIONE

Esistono due tipi di codici:

UN CODICE IN CORPO (o "in linea") è un frammento di codice scritto in linea con il corpo del testo ("incorporato nel testo"), come ad esempio `var i : integer`.

*Codici in corpo e
fuori corpo*

UN CODICE FUORI CORPO (o “in display”) è formato da uno o più capoversi staccati dal testo precedente e seguente mediante spazi di ampiezza adeguata.

La differenza è la stessa che intercorre tra formule matematiche in corpo e fuori corpo.

Per inserire codici in corpo e fuori corpo, L^AT_EX mette a disposizione rispettivamente il comando `\verb` e l’ambiente `verbatim`, che permettono di stampare il codice alla lettera, come se fosse battuto a macchina, con tutti gli spazi e le interruzioni di riga, senza che vengano interpretati i comandi e i caratteri speciali. Il pacchetto `listings` fornisce invece il comando `\lstinline` (per i codici in corpo) e l’ambiente `lstlisting` (per i codici fuori corpo), che permettono di controllare in modo molto preciso il formato del codice stampato; esiste anche il comando `\lstinputlisting`, che consente di stampare il contenuto di un file a sé stante come codice fuori corpo.

Codici in corpo

Il comando `\lstinline`

Ad esempio, il frammento di codice Pascal `var i : integer` si ottiene con l’istruzione `\lstinline!var i : integer!`. Il simbolo `!` è solo un esempio di carattere delimitatore. Si può usare qualsiasi carattere tranne le lettere o lo spazio; `\lstinline$var i : integer$` dà lo stesso risultato. Non è possibile inserire codici in corpo muniti di un riquadro o di uno sfondo colorato.

Codici fuori corpo

L’ambiente `lstlisting`

Il testo che segue è un esempio di scrittura di codice fuori corpo, scritto in linguaggio Pascal, ottenuto inserendo il codice all’interno dell’ambiente `lstlisting`.

```
1 for i:=maxint to 0 do
2 begin
3   { non far nulla }
4 end;
5 write('Benvenuto in Pascal.');
```

Il nome `lstlisting` è stato preferito al più semplice `listing` dal momento che altri pacchetti definiscono già ambienti con quel nome. Per essere compatibile con tali pacchetti, tutti i comandi e gli ambienti di `listings` cominciano con il prefisso `lst`.

Di seguito è riproposto l’esempio precedente: nella parte sinistra vi è il codice sorgente L^AT_EX, e a destra è visualizzato il risultato ottenuto.

```
\begin{lstlisting}
for i:=maxint to 0 do
begin
  { non far nulla }
end;
write('Benvenuto in Pascal.');
```

```
1 for i:=maxint to 0 do
2 begin
3   { non far nulla }
4 end;
5 write('Benvenuto in Pascal.');
```

Codice 1: Un esempio di codice mobile.

```
1 for i:=maxint to 0 do
2 begin
3   { non far nulla }
4 end;
5 write('Benvenuto in Pascal.');
```

Nei paragrafi successivi verrà spiegato come apporre un riquadro come quello dell'esempio in questione e verrà mostrato come colorare le parole e numerare le righe.

L'ambiente `lstlisting`, come del resto il comando `\lstinline`, prevede un parametro opzionale, costituito da un elenco di opzioni, separate da virgole, che specificano particolari caratteristiche del pacchetto. Un'opzione può essere costituita da un'unica parola o da una voce del tipo `<chiave>=<valore>` (i valori `=true` possono essere omessi).

Per esempio, per stampare solo le linee comprese tra la 2 e la 5 si usa il codice:

```
\begin{lstlisting}%
[firstline=2,lastline=5]
for i:=maxint to 0 do
begin
  { non far nulla }
end;
write('Benvenuto in Pascal.');
```

```
2 begin
3   { non far nulla }
4 end;
```

```
\end{lstlisting}
```

L'esempio precedente mostra che le righe vuote alla fine del codice non vengono stampate (la riga 5, nel caso considerato); se si desidera modificare questo comportamento, basta selezionare l'opzione `showlines=true`.

Il comando `\lstinputlisting` stampa il contenuto di un file a sé stante componendolo come codice fuori corpo, senza bisogno di copiarlo fisicamente nel documento. Il comando ha come argomento il nome del file. Di seguito sono riportate le prime righe del file `listings.sty`.

`\lstinputlisting`

```
\lstinputlisting[lastline=4]%
{listings.sty}
```

```
1 %%
2 %% This is file listings.sty,
3 %% generated with
4 %% the docstrip utility.
```

Per evitare problemi è opportuno che il percorso (*path*) dei file inclusi mediante il comando `\lstinputlisting` non contenga spazi.

Codici con didascalia

Il pacchetto `listings` permette di trasformare un codice fuori corpo in un oggetto mobile. Ad esempio, il codice 1 è stato creato con le istruzioni:

Codici mobili

```

\begin{lstlisting}[float=tb,caption={Un esempio di codice mobile.},
                  label=lst:esempio]
for i:=maxint to 0 do
begin
  { non far nulla }
end;
write('Benvenuto in Pascal.');
```

Naturalmente, nel documento è possibile riferirsi al codice tramite il comando `\ref{lst:esempio}`.

Il valore della chiave `float` è un sottoinsieme dell'elenco di caratteri `tbph` e indica a \LaTeX dove posizionare l'oggetto mobile: in cima (*top*) o in fondo (*bottom*) alla pagina (la corrente o la successiva), su una pagina di soli oggetti mobili (*page of floats*) oppure nel punto esatto in cui è situato il relativo ambiente (*here*); la sintassi completa è `float=[*]{< sottoinsieme di tbph >}`, dove l'asterisco opzionale permette di avere un oggetto mobile esteso su tutta la pagina in un documento a più colonne.

È possibile scegliere se posizionare la didascalia sopra (*top*) o sotto (*bottom*) al codice. A tal fine si usa la chiave `captionpos`, che ha come valore una fra le iniziali `t` e `b`.

Codici "fissi"

Il funzionamento delle chiavi `caption` e `label` è simile a quello dei comandi \LaTeX `\caption` e `\label`, con la differenza che `listings` permette di avere codici con didascalia ed etichetta indipendentemente dal fatto che i codici siano mobili:

```

\begin{lstlisting}%
[caption={Un codice "fisso".},
label=lst:fisso]
for i:=maxint to 0 do
begin
  { non far nulla }
end;
write('Benvenuto in Pascal.');
```

Codice 2: Un codice "fisso".

```

1 for i:=maxint to 0 do
2 begin
3   { non far nulla }
4 end;
5 write('Benvenuto in Pascal.);
```

`\lstlistoflistings`

Il comando `\lstlistoflistings` stampa l'elenco dei codici (è analogo a `\listoffigures` e `\listoftables`). Si può specificare una didascalia breve (che verrà stampata nell'elenco dei codici) nel modo seguente:

```
caption={[<didascalia breve>]<didascalia>}
```

Si notino le graffe di raggruppamento. L'opzione `no\ol` fa sì che il relativo codice non compaia nell'elenco dei codici.

Intestazioni dei codici e dell'elenco dei codici in italiano

Se si desidera sostituire le intestazioni (predefinite) "Listing" e "List of Listings" con "Codice" ed "Elenco dei codici" rispettivamente, è sufficiente scrivere (nel preambolo, dopo aver caricato `babel` con l'opzione `italian`):

```

\addto\captionsitalian{%
\renewcommand{\lstlistingname}{Codice}}
```

```
\addto\captionsitalian{%
\renewcommand{\lstlistingname}{Elenco dei codici}}
```

Per avere una semplice didascalia senza l'intestazione "Codice" e senza etichetta, si usa la chiave `title`:

<pre>\begin{lstlisting}[title={Una didascalia senza etichetta.}] for i:=maxint to 0 do begin { non far nulla } end; \end{lstlisting}</pre>	<p>Una didascalia senza etichetta.</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: auto;"> <pre>1 for i:=maxint to 0 do 2 begin 3 { non far nulla } 4 end;</pre> </div>
--	--

2 IMPOSTAZIONI DEL PACCHETTO LISTINGS

Come al solito, il pacchetto si carica con

Le opzioni di listings

```
\usepackage[<opzioni>]{listings}
```

Di regola non c'è bisogno di preoccuparsi di queste opzioni, ma in alcuni casi potrebbero essere utili. Di seguito sono elencate le principali opzioni del pacchetto:

`0.21` permette di compilare documenti scritti per la versione 0.21 di listings;

`draft` non stampa eventuali file ausiliari (può essere utile in fase di correzione delle bozze);

`final` sovrascrive l'eventuale opzione di classe `draft`;

`savemem` risparmia un po' della memoria di L^AT_EX (consente di ridurre il tempo di compilazione di documenti ricchi di codici scritti in linguaggi diversi).

Per specificare le impostazioni, il pacchetto fornisce il fondamentale comando `\lstset` (di regola nel preambolo), che modifica i valori per tutti i codici successivi nello stesso ambiente o gruppo. Il comando ha come argomento un elenco di coppie *<chiave>=<valore>*, separate da virgole.

\lstset

```
\lstset{<chiave>=<valore>,...}
```

In alternativa, è possibile servirsi del parametro opzionale dei comandi `\lstinlne`, `\lstinputlisting` e dell'ambiente `lstlisting`: in questo modo, le opzioni hanno effetto *solo per il relativo codice*.

Per esempio, scrivendo nel preambolo

```
\lstset{basicstyle=\small\ttfamily}
```

tutti i codici del proprio documento verranno stampati in caratteri dal corpo piccolo (scelta peraltro sempre consigliabile, nel caso dei codici fuori corpo), usando il font a spaziatura fissa predefinito, a meno che non si specifichi diversamente per codici particolari.

Tabella 1: Alcuni dei linguaggi predefiniti di listings. I dialetti sottolineati sono predefiniti.

Algol (60, 68)	Ada (2005, 83, 95)
Basic (Visual)	C (ANSI, Objective)
C++ (ANSI, GNU, ISO, Visual)	Cobol (1974, 1985)
Delphi	Fortran (77, 90, 95)
HTML	Java
Lisp	Mathematica (1.0, 3.0, 5.2)
Matlab	METAPOST
Modula-2	Pascal (Borland6, Standard, XSC)
Perl	PHP
PostScript	Prolog
R	SQL
TeX (LaTeX, plain)	XML

Far sì che i codici inseriti con \lstinline siano composti sempre con il corpo corrente

Va notato che il comando `\lstinline` eredita tutte le opzioni impostate con i comandi `\lstset` che lo precedono (nello stesso gruppo). Per esempio, se nel preambolo del documento si seleziona l'opzione globale `basicstyle=\small\ttfamily`, anche `\lstinline` produrrà codici in caratteri dal corpo piccolo. Per far sì che i codici inseriti con `\lstinline` siano composti usando sempre il corpo corrente (evitando di esplicitare ogni volta `basicstyle=\normalsize`), è sufficiente introdurre tutti i codici fuori corpo, con le relative impostazioni, mediante gli ambienti personalizzati descritti nel paragrafo 4 nella pagina 12. In alternativa, è possibile ridefinire il comando `\lstinline` scrivendo nel preambolo le seguenti istruzioni:

```
\makeatletter
\lst@Key{addtobasicstyle}\relax{%
  \toks@=\expandafter{\lst@basicstyle #1}%
  \edef\lst@basicstyle{\the\toks@}
\def\calc@current@size{%
  \edef\current@size{%
    \noexpand\fontsize{\f@size}{\f@baselineskip}%
    \noexpand\selectfont}}

\renewcommand\lstinline[1][]{%
  \leavevmode\bgroup %
  \calc@current@size
  \def\lst@boxpos{b}\lsthk@PreSet
  \lstset{flexiblecolumns,addtobasicstyle=\current@size,#1}%
  \lsthk@TextStyle
  \@ifnextchar\bgroup{\afterassignment\lst@InlineG
  \let\@let@token}\lstinline@}
\makeatother
```

Il pacchetto listings riconosce numerosi linguaggi di programmazione.

Il pacchetto `listings` riconosce un elevato numero di linguaggi di programmazione; nella tabella 1 ne vengono mostrati alcuni, con i loro dialetti (per l'elenco completo dei linguaggi riconosciuti da `listings` si rimanda alla documentazione del pacchetto). Ogni linguaggio è

della forma [*<dialetto>*]*<linguaggio>* e viene selezionato con la chiave language. Senza il parametro opzionale [*<dialetto>*], il pacchetto carica il dialetto predefinito. Ad esempio,

```
\lstset{language=[77]Fortran}
```

seleziona il Fortran 77 e

```
\lstset{language=Pascal}
```

fa lo stesso con il Pascal (standard).

3 PERSONALIZZARE L'ASPETTO DEI CODICI

Il pacchetto listings permette di avere un controllo molto preciso del formato del codice. Negli esempi di codice Pascal proposti fin qui, le parole chiave del linguaggio sono stampate in blu, i commenti in verde e le stringhe in rosso. Nell'esempio seguente, invece, le parole chiave sono in neretto, i commenti in grigio scuro e le stringhe in nero; il font usato è quello a spaziatura fissa.

Il pacchetto listings permette di esercitare un controllo molto preciso sul formato del codice.

```
\begin{lstlisting}
for i:=maxint to 0 do
begin
  { non far nulla }
end;
write('Benvenuto in Pascal.');
```

1	for i:=maxint to 0 do
2	begin
3	{ non far nulla }
4	end;
5	write('Benvenuto_in_Pascal.');

```
\end{lstlisting}
```

Il codice precedente è stato ottenuto specificando le seguenti impostazioni:

```
\lstset{basicstyle=\small\ttfamily,
keywordstyle=\color{black}\bfseries,
commentstyle=\color{darkgray},
stringstyle=\color{black},
showstringspaces=true}
```

L'opzione showstringspaces=true fa sì che gli spazi nelle stringhe siano resi come _ e non come caratteri bianchi.

In generale, è consigliabile scegliere degli stili sobri per i propri codici. Non è possibile fornire indicazioni precise, dal momento che lo stile adottato dipende dal tipo di documento che si sta scrivendo: una presentazione, per esempio, spesso richiede uno stile di maggiore impatto di quello impiegato in un libro. In fin dei conti, si tratta di trovare la giusta misura.

È consigliabile scegliere degli stili sobri per i propri codici.

Di seguito viene presentata una panoramica di strumenti utili per personalizzare i propri codici. Per maggiori dettagli, si rinvia alla ricca documentazione di listings.

3.1 Numerare le righe

È possibile numerare le righe dei codici fuori corpo. Ad esempio, è possibile avere numeri in corpo minuscolo sulla sinistra, una riga sì e una no.

```

\lstset{numbers=left,
        numberstyle=\tiny,
        stepnumber=2}

\begin{lstlisting}
for i:=maxint to 0 do
begin
    { non far nulla }
end;
write('Benvenuto in Pascal.');
```

```

1 for i:=maxint to 0 do
2 begin
3     { non far nulla }
4 end;
5 write('Benvenuto in Pascal.');
```

È possibile interrompere un codice e riprenderlo successivamente.

L'ambiente `lstlisting` permette di interrompere un codice e di riprenderlo successivamente, mantenendo la numerazione corretta.

```

\begin{lstlisting}%
[firstnumber=100]
for i:=maxint to 0 do
begin
    { non far nulla }
end;

\end{lstlisting}
Riprendiamo il codice:
\begin{lstlisting}%
[firstnumber=last]
write('Benvenuto in Pascal.');
```

```

100 for i:=maxint to 0 do
101 begin
102     { non far nulla }
103 end;
```

Riprendiamo il codice:

```

104 write('Benvenuto in Pascal.');
```

In questo esempio, la chiave `firstnumber` è inizialmente impostata a 100; nell'ultima riga della prima parte del codice, il suo valore è `last` (104, nel caso in questione), che permette di continuare la numerazione delle righe nella seconda parte del codice. Va evidenziato che la riga vuota alla fine della prima parte non viene stampata, ma conta ai fini della numerazione. È anche possibile scrivere nel preambolo l'istruzione `\lstset{firstnumber=last}` una volta per tutte, così da avere le righe di tutti i codici del documento numerate consecutivamente (a meno che non si specifichi diversamente per codici particolari).

In alternativa, è possibile dare un nome ad un codice (è necessario distinguere tra lettere maiuscole e minuscole): parti diverse dello stesso codice condividono lo stesso contatore di riga.

```

\begin{lstlisting}[name=Test]
for i:=maxint to 0 do
begin
    { non far nulla }
end;
\end{lstlisting}
Riprendiamo il codice:
\begin{lstlisting}[name=Test]
write('Benvenuto in Pascal.');
```

```

1 for i:=maxint to 0 do
2 begin
3     { non far nulla }
4 end;
```

Riprendiamo il codice:

```

5 write('Benvenuto in Pascal.');
```

Il successivo codice `Test` comincerà con il numero di riga 6, indipendentemente dalla presenza di eventuali altri codici in mezzo.

3.2 Rientri

I rientri (tabulazioni o spazi) servono a rendere il codice più leggibile. Il pacchetto assume che le tabulazioni si arrestino alle colonne 9, 17, 25, 33, e così via. Questi valori dipendono dalla chiave `tabsize`, il cui valore predefinito è 8. Se si modifica tale valore e lo si pone pari ad n , le interruzioni si arresteranno alle colonne $n + 1$, $2n + 1$, $3n + 1$, e così via.

I rientri rendono il codice più leggibile.

```
\lstset{tabsize=2}

\begin{lstlisting}
123456789
  { una tabulazione }
    { due tabulazioni }
123   { 123 + due tab }
\end{lstlisting}
```

1	123456789
2	{ una tabulazione }
3	{ due tabulazioni }
4	123 { 123 + due tab }

Per una migliore comprensione, il codice a destra usa `tabsize=2`, mentre il codice sorgente \LaTeX è impostato con `tabsize=4`.

3.2.1 Rendere visibili le tabulazioni e gli spazi

È possibile rendere visibili i segni di tabulazione e gli spazi. A tal fine, si usano le chiavi `showtabs` e `showspaces`.

```
\lstset{showspaces=true,
        tabsize=8,showtabs=true,
        tab=\rightarrowfill}

\begin{lstlisting}
1  ____for_i:=maxint_to_0_do
2  ____begin
3  ____\rightarrow{non_far_nulla}
4  ____end;
\end{lstlisting}
```

Se si seleziona l'opzione `showspaces` ma non `showtabs`, le tabulazioni sono trasformate in spazi.

3.3 Riquadri

Per munire i propri codici di un riquadro si usa la chiave `frame`, che può assumere come valori `leftline` (che stampa una linea alla sinistra del codice), `topline` (che stampa una linea in cima al codice), `bottomline` (stampa una linea in fondo), `lines` (una linea in cima e una in fondo), `single` (per riquadri singoli) o `shadowbox` (per riquadri ombreggiati); c'è anche `none`, che non stampa alcun riquadro.

È possibile scegliere fra diversi tipi di riquadri.

```
\begin{lstlisting}%
[frame=lines]
1  for i:=maxint to 0 do
2  begin
3  { non far nulla }
4  end;
\end{lstlisting}
```

In alternativa, si possono controllare direttamente le linee in cima (*top*), a destra (*right*), in basso (*bottom*), e a sinistra (*left*) usando rispettivamente le quattro iniziali *t*, *r*, *b* e *l* per ottenere una linea singola e le loro versioni maiuscole per ottenere una linea doppia.

<pre>\begin{lstlisting}[frame=trBL] for i:=maxint to 0 do begin { non far nulla } end; \end{lstlisting}</pre>	<pre>1 for i:=maxint to 0 do 2 begin 3 { non far nulla } 4 end;</pre>
---	---

È possibile avere gli angoli arrotondati (*t*) o dritti (*f*) usando la chiave `frameround`, che ha come valore un elenco di quattro caratteri, scelti fra le lettere *t* o *f*. Il primo carattere è relativo all'angolo in alto a destra, e si continua in senso orario.

<pre>\lstset{frameround=fttt} \begin{lstlisting}[frame=trBL] for i:=maxint to 0 do begin { non far nulla } end; \end{lstlisting}</pre>	<pre>1 for i:=maxint to 0 do 2 begin 3 { non far nulla } 4 end;</pre>
---	---

3.4 Sfondi colorati

Per inserire uno sfondo colorato si usa la chiave `backgroundcolor`, che prende come valore `\color{<colore>}`. È necessario il pacchetto `xcolor` (in alternativa, si può usare il pacchetto `color`).

<pre>\lstset{% backgroundcolor=\color{pink}} \begin{lstlisting}% [frame=single] for i:=maxint to 0 do begin { non far nulla } end; \end{lstlisting}</pre>	<pre>1 for i:=maxint to 0 do 2 begin 3 { non far nulla } 4 end;</pre>
--	---

3.5 Evidenziare parole

*Evidenziare
manualmente alcune
parole*

Di regola, le parole chiave sono evidenziate automaticamente dal pacchetto. Se però si desidera evidenziare manualmente alcune parole, si possono usare le chiavi `emph` e `emphstyle`.

```

\lstset{emph={square, root},
        emphstyle=\bfseries}

\begin{lstlisting}
for i:=maxint to 0 do
begin
    j:=square(root(i));
end;
\end{lstlisting}

```

```

1 for i:=maxint to 0 do
2 begin
3     j:=square(root(i));
4 end;

```

Le chiavi hanno un argomento opzionale che permette di evidenziare parole diverse con stili diversi:

```

\lstset{emph={square},
        emphstyle=\color{Green},
        emph={[2]root},
        emphstyle={[2]\color{Red}}}

\begin{lstlisting}
for i:=maxint to 0 do
begin
    j:=square(root(i));
end;
\end{lstlisting}

```

```

1 for i:=maxint to 0 do
2 begin
3     j:=square(root(i));
4 end;

```

I colori definiti con `\color{Green}` e `\color{Red}` richiedono il pacchetto `xcolor` con l'opzione `dvipsnames`.

3.6 Inserire una parola nell'indice analitico

Per inserire una parola nell'indice analitico si usa la chiave `index`:

```

\lstset{index={square, root}}

\begin{lstlisting}
for i:=maxint to 0 do
begin
    j:=square(root(i));
end;
\end{lstlisting}

```

```

1 for i:=maxint to 0 do
2 begin
3     j:=square(root(i));
4 end;

```

In questo modo, le voci `square` e `root` vengono inserite nell'indice analitico.

3.7 L'allineamento delle colonne

Uno degli aspetti tipografici che più balzano all'occhio nella composizione dei codici, oltre ai colori, i rientri e i riquadri, è l'allineamento fra le colonne.

Di regola, nella composizione dei codici si desidera mantenere l'allineamento delle colonne: è necessario allora impiegare un font a spaziatura fissa. Se però l'allineamento non ha funzioni sintattiche, si può usare il font che si vuole. In entrambi i casi è conveniente

Di regola, nella composizione dei codici si usa un font a spaziatura fissa.

selezionare l'opzione `columns=fullflexible`, che fa sì che tutti i caratteri siano composti con la loro larghezza "naturale". (L'opzione `columns=fixed`, sconsigliata, cerca invece di mantenere l'allineamento tra le colonne modificando la spaziatura tra i font, con risultati tipografici decisamente discutibili.)

Si consiglia inoltre di scegliere sempre l'opzione `keepspace=true`, che impedisce che vengano modificati gli spazi tra le parole nel tentativo di mantenere l'allineamento tra le colonne.

4 TECNICHE AVANZATE

4.1 Ambienti personalizzati

*Il pacchetto listings
permette di definire
ambienti
personalizzati.*

Il pacchetto `listings` permette di definire ambienti personalizzati mediante i quali introdurre i propri codici. A tal fine si utilizza il comando `\lstnewenvironment`, la cui sintassi è analoga a quella del comando \LaTeX `\newenvironment`.

```
\lstnewenvironment{<nome>}[<numero di argomenti>][<predefinito>]
  {<prima>}{<dopo>}
```

Quando \LaTeX incontra `\begin{<nome>}`, compone un codice fuori corpo con le impostazioni specificate nell'argomento `<prima>`, e quando incontra `\end{<nome>}`, inserisce quanto specificato nell'argomento `<dopo>`. L'ambiente che viene definito può ricevere (in apertura) un certo `<numero di argomenti>`, il primo dei quali può essere facoltativo, se viene espressa anche la seconda opzione di `\lstnewenvironment`. Se nell'uso di un nuovo ambiente che accetta un argomento facoltativo quest'ultimo non viene specificato, esso assume il valore `<predefinito>`.

Il seguente esempio mostra l'uso del comando `\lstnewenvironment`. Se scriviamo nel preambolo

```
\lstnewenvironment{pascal}{\lstset{language=pascal}}{}
```

l'ambiente `pascal` si usa nel modo seguente:

<pre>\begin{pascal} for i:=maxint to 0 do begin { non far nulla } end; write('Benvenuto in Pascal. '); \end{pascal}</pre>	<pre>1 for i:=maxint to 0 do 2 begin 3 { non far nulla } 4 end; 5 write('Benvenuto in Pascal.');</pre>
---	--

Gli ambienti con un argomento facoltativo permettono di specificare delle impostazioni differenti per ciascun codice:

```
\lstnewenvironment{pascalx}[1][
  {\lstset{language=pascal,numbers=left,float=tb,#1}}{}
```

4.2 Inserire comandi \LaTeX all'interno di un codice

Normalmente, il codice contenuto all'interno dei comandi e degli ambienti di listings viene stampato alla lettera, senza che \LaTeX interpreti i comandi e i caratteri speciali.

Tuttavia, a volte si può voler inserire comandi \LaTeX all'interno di un codice fuori corpo, in modo che essi vengano interpretati. A tal fine è necessario specificare una sequenza di caratteri che permetta di "passare a \LaTeX " e successivamente di "ritornare al codice". Ad esempio:

È possibile inserire comandi \LaTeX all'interno di un codice fuori corpo.

```
\lstset{escapeinside={f!}{!f}}
```

Se all'interno di un codice si scrive `f!` si passa a \LaTeX ; scrivendo `!f` si ritorna al codice.

Si noti che ogni passaggio a \LaTeX può disturbare l'allineamento delle colonne, dal momento che il pacchetto non ha modo di controllare la spaziatura una volta passati a \LaTeX . Il seguente esempio è in C++:

```
\lstset{language=C++,
commentstyle=\color{black}}

\begin{lstlisting}
// Calcoliamo  $a_{ij}$ 
1 A[i][j] = A[j][j]/A[i][j];
2
\end{lstlisting}
```

4.3 Definire nuovi linguaggi

Il pacchetto permette di definire linguaggi personalizzati, specificando il formato delle parole chiavi, dei commenti e delle stringhe.

Si possono definire linguaggi personalizzati.

```
\lstdefinlanguage{rock}
{morekeywords=%
one,two,three,four,five,six,seven,eight,nine,ten,eleven,
twelve,o,clock,rock,around,the,tonight},
morecomment=[l][\color{Green}]{//},
morecomment=[s][\color{Green}]{/*}{*/},
morestring=[b][\color{Red}]}
```

La chiave `morekeywords` permette di aggiungere quante parole chiave si vuole ad un linguaggio.

Nuove parole chiave

Per quanto riguarda i commenti e le stringhe, si usano le parole chiave `morecomment` e `morestring`, rispettivamente. Il loro valore comincia con un parametro [*tipo*], che indica il tipo di commento o di stringa, seguito da un altro parametro [*stile*], che indica lo stile del commento o della stringa, e da numero variabile di delimitatori di apertura e di chiusura.

Nuovi commenti

I tipi più comuni per i commenti sono `l` e `s`, rispettivamente con uno e due delimitatori. Un commento di tipo `l` inizia in un punto qualunque della riga e si estende fino alla fine della riga stessa, mentre un commento di tipo `s` inizia con il primo delimitatore e si conclude con il secondo delimitatore.

Il tipo più comune per le stringhe è `b`, con un delimitatore che inizia

Nuove stringhe

e conclude la stringa; se dentro una stringa c'è bisogno di scrivere il delimitatore (caso raro, ma possibile), si può usare il carattere di barra rovescia (*backslash*) per scrivere il delimitatore senza che venga conclusa la stringa.

```
\begin{lstlisting}
Parole chiave: one, two, ...
"Una str\"inga" Del testo
// Una riga di commento
/*Un commento*/ Del testo
\end{lstlisting}
```

1 Parole chiave: one, two, ...
2 "Una str\"inga" Del testo
3 // Una riga di commento
4 /*Un commento*/ Del testo

Delimitatori di tipo
generale

Per l'elenco completo dei tipi di commento e di stringa definiti da listings si rimanda alla documentazione del pacchetto.

Si possono infine definire delimitatori più generali con la chiave `moredelim`. I tipi possibili sono `l` e `s`, che possono essere preceduti dalla lettera `i` (*invisible*), che elimina i delimitatori dal documento finale. Marcatore di questo tipo consentono di impostare stili a piacere. Ad esempio, se si scrive nel preambolo

```
\lstset{moredelim=[is][\color{orange}]{|*}{*|}}
```

i delimitatori `|* e *|` si usano nel modo seguente:

```
\begin{lstlisting}
Una parola |*arancione*|.
\end{lstlisting}
```

1 Una parola arancione.

RIFERIMENTI BIBLIOGRAFICI

BROOKS, M. (2008), *The listings package*, Manuale d'uso del pacchetto listings, <http://www.ctan.org/tex-archive/macros/latex/contrib/listings/listings.pdf>.

PANTIERI, L. (2008), *L'arte di scrivere con L^AT_EX*, http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf.