

L^AT_EX Kurs

Code Darstellung in L^AT_EX

Sascha Frank
<http://www.latex-kurs.de/kurse/kurse.html>

Übersicht

Basis L^AT_EX

Verbatim

Pseudocode

Quellcode

Bisher

Befehl

`\verb`

Beispiel

`\verb+\LaTeX+ → \LaTeX`

Umgebung

`verbatim` und `verbatim*`

Beispiele

```
\begin{verbatim}
```

Das `\LaTeX` Symbol

```
\end{verbatim}
```

Das `\LaTeX` Symbol

```
\begin{verbatim*}
```

Das `\LaTeX` Symbol

```
\end{verbatim*}
```

Das `\LaTeX` Symbol

verbatim Paket

Paket

```
\usepackage{verbatim}
```

Umgebungen

Verbesserte Versionen von `verbatim` und `verbatim*`

Kommentarumgebung `comment`

Befehle

`verbatiminput` und `verbatiminput*`

Umgebungen

verbatim und verbatim*

Wie bisher, aber bei älteren Paketversion wird alles was hinter `\end{verbatim}` beziehungsweise hinter `\end{verbatim*}` in der Zeile steht ignoriert.

comment

Kommentarumgebung um mehrzeilige Bereiche auszukomentieren.

```
\begin{comment}
```

Der Text der innerhalb der `\emph{comment}` Umgebung steht wird nicht ausgegeben und es werden auch keine Befehle abgearbeitet.

```
\end{comment}
```

algorithms Bundle

algorithms Bundle

Besteht aus zwei Paketen:
algorithmic – zum Setzen des Pseudocodes
algorithm – Gleitobjekt

einbinden

```
\usepackage{algorithmic}  
\usepackage{algorithm}
```

Umgebung

Heißen wie die Pakete

algorithmic

algorithmic Umgebung

```
\begin{algorithmic}[Schrittweite]
```

```
...
```

```
\end{algorithmic}
```

ohne Nummerierung

Einfach die Option weglassen

algorithmic Befehle

Schleifen

```
\LOOP \STATE \ENDLOOP, \FOR, \WHILE, etc.
```

Fallunterscheidung

```
\IF ... \ELSE ... \ENDIF, etc.
```

Textausgabe

```
\COMMENT, \RETURN, \PRINT, etc.
```

logische Ausdrücke

```
\AND, \OR, ..., \TRUE, \FALSE
```

algorithmic Beispiel

```
\begin{algorithmic}
\IF{$a = 0$}
\RETURN $b$
\ELSE
\WHILE{$b \neq 0$}
\IF{$a > b$}
\STATE $a \leftarrow a - b$
\ELSE
\STATE $b \leftarrow b - a$
\ENDIF
\ENDWHILE
\RETURN $a$
\ENDIF
\end{algorithmic}
```

if $a = 0$ **then**
 return b
else
 while $b \neq 0$ **do**
 if $a > b$ **then**
 $a \leftarrow a - b$
 else
 $b \leftarrow b - a$
 end if
 end while
return a
end if

algorithm

Darstellung

plain, ruled, boxed

Zählung

part, chapter, ..., nothing

Beschriftung

Mit `\floatname{algorithm}{Neue Beschriftung}` lässt sich der Bezeichner der Beschriftung ändern.

Übersicht einfügen

Mit `\listofalgorithms` wird die Übersicht eingefügt.

Umbenennen geht mit:

```
\renewcommand{\listalgorithmname}{Pseudocode Liste}
```

```
...
\usepackage[ruled]{algorithm}
...
\begin{algorithm}
\caption{Euclidean algorithm}
\begin{algorithmic}
\IF{$a = 0$}
\RETURN $b$
\ELSE
\WHILE{$b \neq 0$}
\IF{$a > b$}
\STATE $a \leftarrow a - b$
\ELSE
\STATE $b \leftarrow b - a$
\ENDIF
\ENDWHILE
\RETURN $a$
\ENDIF
\end{algorithmic}
\end{algorithm}
...
```

Fortsetzung

Algorithm 1 Euclidean algorithm

```
if  $a = 0$  then
    return  $b$ 
else
    while  $b \neq 0$  do
        if  $a > b$  then
             $a \leftarrow a - b$ 
        else
             $b \leftarrow b - a$ 
        end if
    end while
    return  $a$ 
end if
```

minted

Paket

```
\usepackage{minted}
```

- ▶ über 300 Programmiersprachen
- ▶ neue Umgebung
- ▶ neue Befehle
- ▶ viele Optionen

Teil der Optionen

Rücksetzpunkt

Nummerierung kann mit `chapter` bzw. `section` auf Kapitel- bzw. Abschnittsweise Zählung verändert werden.

listings

Mit `newfloat = true` wird das `newfloat` anstelle des `float` Paketes zur Erstellung der `listing` Umgebung verwendet.

Codezeilen Nummerierung

Fortlaufenden Nummerierung über Programmiersprachengrenzen hinaus: `langlinenos = true`

minted Umgebung

```
\begin{minted}[Optionen / Einstellungen]{Sprache}  
CODE in der SPRACHE  
    CODE ...  
\end{minted}
```

Optionen

- ▶ Nummerierung Ob / Wo
- ▶ Schrift Art / Größe
- ▶ Farbe Hintergrund / Style
- ▶ Umbruch Innerhalb / Leerzeichen
- ▶ ...

Befehle

mint

Bindet Code Stücke ein, davor und danach `auto.` Zeilenumbruch.

```
\mint[Optionen]{Sprache}{CODE in der SPRACHE}
```

mintinline

Bindet Code Stücke ein, ohne `auto.` Zeilenumbruch.

```
\mintinline[Optionen]{Sprache}{CODE in der SPRACHE}
```

inputminted

Bindet ganze Dateien mit Code ein.

```
\inputminted[Optionen]{Sprache}{Dateiname}
```

Beispiele

Text vor dem mint Befehl `\mint{python}{print(x**2)}` und Text danach.

Text vor dem mint Befehl

```
print(x**2)
```

und Text danach.

Der mintinline Befehl `\mintinline{python}{print(x**2)}` und Text danach.

Der mintinline Befehl `print(x**2)` und Text danach.

```
\inputminted{c}{hello.c}
```

```
printf("Hello World");
```

listings

```
\begin{listing}[H]
\begin{minted}[style=trac]{c}
int main() {
    printf("Hello World");
    return 0;
}
\end{minted}
\caption{Ein Beispiel für Hello World.}
\label{Beispiel}
\end{listing}
```

Einfügen

Mit `\listoflistings` wird das Verzeichnis an die gewünschte Stelle im Dokument gesetzt.

```
int main() {
    printf("Hello World");
    return 0;
}
```

[Listing 1](#): Ein Beispiel für Hello World.

Hinweise

Beim Kompilieren ist in der Regel ein Shell escape zu setzen, das bedeutet:

```
pdflatex -shell-escape Datei.tex
```

Um herauszufinden welche Sprachen unterstützt werden den folgenden Befehl in der Konsole ausführen.

```
pygmentize -L lexers
```

Nur bedingt für beamer class geeignet...