# ZfS–Kurs „LaTeX“

Karin Halupczok
Email: Karin.Halupczok@math.uni-freiburg.de

WiSe 2009/2010

Erste Zusatzthemensitzung:
Eigene Strukturen

http://home.mathematik.uni-freiburg.de/halupczok/latex2.html

Eigene Längen

Eigene Zähler

Eigene Befehle

Eigene Umgebungen

# Eigene Längenmaße

Mit

```
\newlength{\meineLaenge}
```

wird ein neuer Längenbefehl definiert. Standardmäßig bekommt er so die Länge Null zugewiesen.

# Eigene Längenmaße

Mit

```
\newlength{\meineLaenge}
```

wird ein neuer Längenbefehl definiert. Standardmäßig bekommt er so die Länge Null zugewiesen.

Auch für selbstdefinierte Längen gibt es die Befehle: `settowidth`, `setlength`, `addtolength`

# Eigene Längenmaße

Mit

```
\newlength{\meineLaenge}
```

wird ein neuer Längenbefehl definiert. Standardmäßig bekommt er so die Länge Null zugewiesen.

Auch für selbstdefinierte Längen gibt es die Befehle: `settowidth`, `setlength`, `addtolength`

Den Betrag einer Länge kann man mit `\the\meineLaenge` in der Längeneinheit pt ausgeben lassen.

# Eigene Längenmaße

Mit

```
\newlength{\meineLaenge}
```

wird ein neuer Längenbefehl definiert. Standardmäßig bekommt er so die Länge Null zugewiesen.

Auch für selbstdefinierte Längen gibt es die Befehle: `settowidth`, `setlength`, `addtolength`

Den Betrag einer Länge kann man mit `\the\meineLaenge` in der Längeneinheit pt ausgeben lassen.

Umrechnen: 72.27pt = 1in bzw. 2.84528pt = 1mm

Eigene Längen

Eigene Zähler

Eigene Befehle

Eigene Umgebungen

# Definition eigener Zähler

Mit

```
\newcounter{meinZaehler}[Ruecksetzzaehler]
```

wird ein eigener Zähler namens *meinZaehler* eingerichtet.

# Definition eigener Zähler

Mit

```
\newcounter{meinZaehler}[Ruecksetzzaehler]
```

wird ein eigener Zähler namens *meinZaehler* eingerichtet.

Die eigenen Zähler können so wie die in LaTeX vordefinierten Zähler verändert werden: mit
`\setcounter`, `\addtocounter`, `\stepcounter`...

# Definition eigener Zähler

Mit

```
\newcounter{meinZaehler}[Ruecksetzzaehler]
```

wird ein eigener Zähler namens *meinZaehler* eingerichtet.

Die eigenen Zähler können so wie die in LaTeX vordefinierten Zähler verändert werden: mit
`\setcounter`, `\addtocounter`, `\stepcounter`...

Ein Zähler mit dem Namen *meinZaehler* wird ausgegeben mit `\themeinZaehler`.

# Eigene Zähler

Der Befehl `\stepcounter{meinZaehler}`

erhöht also nicht nur den Zähler *meinZaehler* um eins, sondern setzt außerdem auch alle die Zähler auf Null zurück, für die *meinZaehler* als Rücksetzer definiert wurde.

# Eigene Zähler

Der Befehl `\stepcounter{meinZaehler}`

erhöht also nicht nur den Zähler *meinZaehler* um eins, sondern setzt außerdem auch alle die Zähler auf Null zurück, für die *meinZaehler* als Rücksetzer definiert wurde.

Sofern also

```
\newcounter{Zaehler}[meinZaehler]
```

definiert wurde, wird der Zähler *Zaehler* nach dem `stepcounter{meinZaehler}`-Befehl auf Null stehen.

# Eigene Zähler

Der Befehl `\stepcounter{meinZaehler}`

erhöht also nicht nur den Zähler *meinZaehler* um eins, sondern setzt außerdem auch alle die Zähler auf Null zurück, für die *meinZaehler* als Rücksetzer definiert wurde.

Sofern also

```
\newcounter{Zaehler}[meinZaehler]
```

definiert wurde, wird der Zähler *Zaehler* nach dem `stepcounter{meinZaehler}`-Befehl auf Null stehen.

Ebenso funktioniert

```
\refstepcounter{meinZaehler}
```

wobei dieser Befehl darüberhinaus es ermöglicht, daß auf den Zählerstand mit Verweisen Bezug genommen werden kann (also mit `\ref{Marke}`, falls nach diesem Befehl die Marke `\label{Marke}` gesetzt wurde).

# Ein Beispiel

**Beispiel:**
Dieser$^{(1)}$ Satz$^{(2)}$ hat$^{(3)}$ fünf$^{(4)}$ Wörter$^{(5)}$.
Das Verb steht in diesem Satz an 3. Stelle.

# Ein Beispiel

**Beispiel:**
Dieser$^{(1)}$ Satz$^{(2)}$ hat$^{(3)}$ fünf$^{(4)}$ Wörter$^{(5)}$.
Das Verb steht in diesem Satz an 3. Stelle.

Dies wird so gemacht:

```
\newcommand{\wortnummer}%
{\refstepcounter{Wortzaehler}$^{(\theWortzaehler)}$}

Dieser\wortnummer\
Satz\wortnummer\
hat\wortnummer\
\label{Verb}
fünf\wortnummer\
Wörter\wortnummer.

Das Verb steht in diesem Satz an \ref{Verb}.\ Stelle.
```

Eigene Längen

Eigene Zähler

Eigene Befehle

Eigene Umgebungen

# Eigene Befehle definieren

Mit

```
\newcommand{\meinBefehl}[n]{Definition}
```

wird ein neuer Befehl definiert. Damit kann man häufig benutzte Befehle abkürzen. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9);

# Eigene Befehle definieren

Mit

```
\newcommand{\meinBefehl}[n]{Definition}
```

wird ein neuer Befehl definiert. Damit kann man häufig benutzte Befehle abkürzen. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

# Eigene Befehle definieren

Mit

```
\newcommand{\meinBefehl}[n]{Definition}
```

wird ein neuer Befehl definiert. Damit kann man häufig benutzte Befehle abkürzen. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

**Beispiel 1:** für einen Befehl ohne Parameter:

```
\newcommand{\Z}{\vspace{\baselineskip}}
```

# Eigene Befehle definieren

Mit

```
\newcommand{\meinBefehl}[n]{Definition}
```

wird ein neuer Befehl definiert. Damit kann man häufig benutzte Befehle abkürzen. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

**Beispiel 1:** für einen Befehl ohne Parameter:

```
\newcommand{\Z}{\vspace{\baselineskip}}
```

Der Befehl `\Z` erzeugt dann eine Leerzeile Platz.

# Eigene Befehle definieren

Mit

```
\newcommand{\meinBefehl}[n]{Definition}
```

wird ein neuer Befehl definiert. Damit kann man häufig benutzte Befehle abkürzen. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

**Beispiel 1:** für einen Befehl ohne Parameter:

```
\newcommand{\Z}{\vspace{\baselineskip}}
```

Der Befehl `\Z` erzeugt dann eine Leerzeile Platz.

**Beispiel 2:** Abkürzung für den Folgepfeil:

```
\newcommand{\f}{\Rightarrow}
```

# Eigene Befehle definieren

Mit

```
\newcommand{\meinBefehl}[n]{Definition}
```

wird ein neuer Befehl definiert. Damit kann man häufig benutzte Befehle abkürzen. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

**Beispiel 1:** für einen Befehl ohne Parameter:

```
\newcommand{\Z}{\vspace{\baselineskip}}
```

Der Befehl `\Z` erzeugt dann eine Leerzeile Platz.

**Beispiel 2:** Abkürzung für den Folgepfeil:

```
\newcommand{\f}{\Rightarrow}
```

Der Befehl `$a\f b$` ergibt

# Eigene Befehle definieren

Mit

```
\newcommand{\meinBefehl}[n]{Definition}
```

wird ein neuer Befehl definiert. Damit kann man häufig benutzte Befehle abkürzen. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

**Beispiel 1:** für einen Befehl ohne Parameter:

```
\newcommand{\Z}{\vspace{\baselineskip}}
```

Der Befehl `\Z` erzeugt dann eine Leerzeile Platz.

**Beispiel 2:** Abkürzung für den Folgepfeil:

```
\newcommand{\f}{\Rightarrow}
```

Der Befehl `$a\f b$` ergibt $a \Rightarrow b$.

# Eigene Befehle definieren

Mit

```
\newcommand{\meinBefehl}[n]{Definition}
```

wird ein neuer Befehl definiert. Damit kann man häufig benutzte Befehle abkürzen. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

**Beispiel 1:** für einen Befehl ohne Parameter:

```
\newcommand{\Z}{\vspace{\baselineskip}}
```

Der Befehl `\Z` erzeugt dann eine Leerzeile Platz.

**Beispiel 2:** Abkürzung für den Folgepfeil:

```
\newcommand{\f}{\Rightarrow}
```

Der Befehl `$a\f b$` ergibt $a \Rightarrow b$.

Existiert ein Befehl schon, kann man ihn mit `\renewcommand` statt `\newcommand` umdefinieren.

# Eigene Befehle mit Parametern

Die Parameter werden in der Befehlsdefinition mit #1 bis #9 angesprochen.

# Eigene Befehle mit Parametern

Die Parameter werden in der Befehlsdefinition mit #1 bis #9 angesprochen.

**Beispiel** für einen Befehl mit drei Parametern:

# Eigene Befehle mit Parametern

Die Parameter werden in der Befehlsdefinition mit #1 bis #9 angesprochen.

**Beispiel** für einen Befehl mit drei Parametern:

```
\newcommand{\adresse}[3]{%
\fbox{\begin{minipage}[t]{0.3\textwidth}%
#1 \\ #2 \\ #3\end{minipage}}}
```

# Eigene Befehle mit Parametern

Die Parameter werden in der Befehlsdefinition mit #1 bis #9 angesprochen.

**Beispiel** für einen Befehl mit drei Parametern:

```
\newcommand{\adresse}[3]{%
\fbox{\begin{minipage}[t]{0.3\textwidth}%
#1 \\ #2 \\ #3\end{minipage}}}
```

Die Eingabe von
`\adresse{Math.~Institut}{Eckerstr.~1}{79104~Freiburg}`
ergibt dann

# Eigene Befehle mit Parametern

Die Parameter werden in der Befehlsdefinition mit #1 bis #9 angesprochen.

**Beispiel** für einen Befehl mit drei Parametern:

```
\newcommand{\adresse}[3]{%
\fbox{\begin{minipage}[t]{0.3\textwidth}%
#1 \\ #2 \\ #3\end{minipage}}}
```

Die Eingabe von
`\adresse{Math.~Institut}{Eckerstr.~1}{79104~Freiburg}`
ergibt dann

Math. Institut
Eckerstr. 1
79104 Freiburg

# Befehl mit optionalem Parameter

Der erste Parameter kann vordefiniert werden und wird dann zu einem optionalen Parameter.

# Befehl mit optionalem Parameter

Der erste Parameter kann vordefiniert werden und wird dann zu einem optionalen Parameter.

**Beispiel:**

```
\newcommand{\name}[3][Freiburg]{#2 #3 aus #1}
```

# Befehl mit optionalem Parameter

Der erste Parameter kann vordefiniert werden und wird dann zu einem optionalen Parameter.

**Beispiel:**

```
\newcommand{\name}[3][Freiburg]{#2 #3 aus #1}
```

`\name{Herr}{Maier}` ergibt dann

# Befehl mit optionalem Parameter

Der erste Parameter kann vordefiniert werden und wird dann zu einem optionalen Parameter.

**Beispiel:**

```
\newcommand{\name}[3][Freiburg]{#2 #3 aus #1}
```

`\name{Herr}{Maier}` ergibt dann Herr Maier aus Freiburg.

# Befehl mit optionalem Parameter

Der erste Parameter kann vordefiniert werden und wird dann zu einem optionalen Parameter.

**Beispiel:**

```
\newcommand{\name}[3][Freiburg]{#2 #3 aus #1}
```

`\name{Herr}{Maier}` ergibt dann Herr Maier aus Freiburg.
`\name[Wuppertal]{Frau}{Schmidt}` ergibt aber

# Befehl mit optionalem Parameter

Der erste Parameter kann vordefiniert werden und wird dann zu einem optionalen Parameter.

**Beispiel:**

```
\newcommand{\name}[3][Freiburg]{#2 #3 aus #1}
```

`\name{Herr}{Maier}` ergibt dann Herr Maier aus Freiburg.
`\name[Wuppertal]{Frau}{Schmidt}` ergibt aber
Frau Schmidt aus Wuppertal.

# Befehl mit optionalem Parameter

Der erste Parameter kann vordefiniert werden und wird dann zu einem optionalen Parameter.

**Beispiel:**

```
\newcommand{\name}[3][Freiburg]{#2 #3 aus #1}
```

`\name{Herr}{Maier}` ergibt dann Herr Maier aus Freiburg.
`\name[Wuppertal]{Frau}{Schmidt}` ergibt aber
Frau Schmidt aus Wuppertal.

Der Befehl hat für den ersten Parameter also eine Standardeinstellung (hier der Text „Freiburg“), den man optional verändern kann.

# Befehl mit optionalem Parameter

Der erste Parameter kann vordefiniert werden und wird dann zu einem optionalen Parameter.

**Beispiel:**

```
\newcommand{\name}[3][Freiburg]{#2 #3 aus #1}
```

`\name{Herr}{Maier}` ergibt dann Herr Maier aus Freiburg.
`\name[Wuppertal]{Frau}{Schmidt}` ergibt aber Frau Schmidt aus Wuppertal.

Der Befehl hat für den ersten Parameter also eine Standardeinstellung (hier der Text „Freiburg“), den man optional verändern kann.

*Bemerkung:* Soll innerhalb der Befehlsdefinition eine Matheumgebung stehen, nimmt man dafür besser `\ensuremath{`*Formel*`}` anstatt die Formel darin mit `$`*Formel*`$` zu umschließen.

Eigene Längen

Eigene Zähler

Eigene Befehle

Eigene Umgebungen

# Eigene Umgebungen definieren

`\newenvironment{meineUmgebung}[n]{AnfangDef}{EndeDef}`

definiert eine neue Umgebung. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9);

# Eigene Umgebungen definieren

`\newenvironment{meineUmgebung}[n]{AnfangDef}{EndeDef}`

definiert eine neue Umgebung. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

# Eigene Umgebungen definieren

`\newenvironment{meineUmgebung}[n]{AnfangDef}{EndeDef}`

definiert eine neue Umgebung. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

Üblicherweise greift man bei der Definition auf eine vorprogrammierte Umgebung zurück.

# Eigene Umgebungen definieren

`\newenvironment{meineUmgebung}[n]{AnfangDef}{EndeDef}`

definiert eine neue Umgebung. Dabei ist n die Anzahl der Parameter (zwischen 0 und 9); `[0]` kann entfallen.

Üblicherweise greift man bei der Definition auf eine vorprogrammierte Umgebung zurück.

Parameter dürfen nur im `{AnfangDef}`-Teil der Definition auftreten.

# Beispiel für eine eigene Umgebungen

**Beispiel:**

```
\newenvironment{xxx}[1]
   {\hspace*{#1cm}\begin{minipage}[b]{5cm}\itshape}
   {\end{minipage}ENDE}
```

# Beispiel für eine eigene Umgebungen

**Beispiel:**

```
\newenvironment{xxx}[1]
   {\hspace*{#1cm}\begin{minipage}[b]{5cm}\itshape}
   {\end{minipage}ENDE}

\begin{xxx}{2}
hier kommt Text, viel Text, und noch mehr Text
\end{xxx}
```

ergibt

# Beispiel für eine eigene Umgebungen

**Beispiel:**

```
\newenvironment{xxx}[1]
   {\hspace*{#1cm}\begin{minipage}[b]{5cm}\itshape}
   {\end{minipage}ENDE}

\begin{xxx}{2}
hier kommt Text, viel Text, und noch mehr Text
\end{xxx}
```

ergibt

*hier kommt Text, viel Text, und noch mehr Text* ENDE

# Beispiel für eine eigene Listenumgebung

Es soll eine Listenumgebung `Liste` mit einem Paramater definiert werden. Diese soll wie die `itemize`–Umgebung funktionieren, der Parameter bestimmt dabei den Abstand (in mm) zwischen den einzelnen Punkten.

# Beispiel für eine eigene Listenumgebung

Es soll eine Listenumgebung `Liste` mit einem Paramater definiert werden. Diese soll wie die `itemize`–Umgebung funktionieren, der Parameter bestimmt dabei den Abstand (in mm) zwischen den einzelnen Punkten.

In der Anwendung soll der Quellcode dann so aussehen:

```
\begin{Liste}{4}
\item  erster Stichpunkttext
\item  zweiter Stichpunkttext
\end{Liste}
```

# Beispiel für eine eigene Listenumgebung

Es soll eine Listenumgebung `Liste` mit einem Paramater definiert werden. Diese soll wie die `itemize`–Umgebung funktionieren, der Parameter bestimmt dabei den Abstand (in mm) zwischen den einzelnen Punkten.

In der Anwendung soll der Quellcode dann so aussehen:

```
\begin{Liste}{4}
\item  erster Stichpunkttext
\item  zweiter Stichpunkttext
\end{Liste}
```

Es soll also innerhalb der Liste die Länge `\itemsep` auf #1mm gestellt werden.

# Lösung

Die Liste wird wie folgt definiert.

```
\newenvironment{Liste}[1]
   {\begin{itemize}\setlength{\itemsep}{#1mm}}
   {\end{itemize}}
```

# Lösung

Die Liste wird wie folgt definiert.

```
\newenvironment{Liste}[1]
   {\begin{itemize}\setlength{\itemsep}{#1mm}}
   {\end{itemize}}
```

So können also Stichpunktlisten mit beliebigem Abstand zwischen den Listenpunkten erzeugt werden.

# Ein weiteres Listenbeispiel

Ganz selbstprogrammierte Listen können durch Verwenden der Umgebung `list` definiert werden wie in diesem Beispiel:

```
\newcounter{itemzaehler}
\newenvironment{MList}[1][\alph]
{\begin{list}{\textbf(#1{itemzaehler})}
                {\usecounter{itemzaehler}}}
{\end{list}}
```

# Ein weiteres Listenbeispiel

Ganz selbstprogrammierte Listen können durch Verwenden der Umgebung `list` definiert werden wie in diesem Beispiel:

```
\newcounter{itemzaehler}
\newenvironment{MList}[1][\alph]
{\begin{list}{\textbf(#1{itemzaehler})}
                {\usecounter{itemzaehler}}}
{\end{list}}
```

Bemerkung: Der erste Parameter von `list` definiert dabei die Standardmarke der Liste (mit der durchnumeriert wird, hier mit dem selbstdefinierten Zähler `itemzaehler`), der zweite enthält Befehle, die noch vor dem Listensatz ausgeführt werden.

# Verwendung der Liste

Die Liste wird wie folgt angewandt:

```
\begin{MList}
  \item erster Versuch
  \item ein zweiter
\end{MList}
```

# Verwendung der Liste

Die Liste wird wie folgt angewandt:

```
\begin{MList}
  \item erster Versuch
  \item ein zweiter
\end{MList}
```

Das Ergebnis ist:

(a) erster Versuch

(b) ein zweiter

# Verwendung der Liste

Die Liste wird wie folgt angewandt:

```
\begin{MList}
  \item erster Versuch
  \item ein zweiter
\end{MList}
```

Das Ergebnis ist:

(a) erster Versuch

(b) ein zweiter

Als optionaler Parameter der Liste kann man die Darstellungsart des Zählers wählen, voreingestellt ist `\alph` dafür.

Somit ergibt die Eingabe von

```
\begin{MList}[\arabic]
  \item erster Versuch
  \item ein zweiter
\end{MList}
```

# Verwendung der Liste

Die Liste wird wie folgt angewandt:

```
\begin{MList}
  \item erster Versuch
  \item ein zweiter
\end{MList}
```

Das Ergebnis ist:

(a) erster Versuch

(b) ein zweiter

Als optionaler Parameter der Liste kann man die Darstellungsart des Zählers wählen, voreingestellt ist `\alph` dafür.

Somit ergibt die Eingabe von

```
\begin{MList}[\arabic]
  \item erster Versuch
  \item ein zweiter
\end{MList}
```

das Ergebnis:

(1) erster Versuch

(2) ein zweiter