

# **Funktionale Programmierung**

## **Übung 01**

Dozent: mein Dozent

Tutor: mein Tutor

Studenten: person1 und person2

tt. mm. jjjj

# Inhaltsverzeichnis

<b>1</b>	<b>Text schreiben</b>	<b>3</b>
<b>2</b>	<b>Tabulatoren</b>	<b>3</b>
2.1	Tabulatoren zur Textstrukturierung . . . . .	3
2.2	Tabellen mittels Tabulatoren . . . . .	4
<b>3</b>	<b>Formeln</b>	<b>4</b>
3.1	Formeln im Text . . . . .	4
3.2	Formeln im abgesetzten Mathematikmodus . . . . .	4
3.2.1	Gleichungen . . . . .	5
<b>4</b>	<b>Quellcode</b>	<b>5</b>
4.1	Quellcode inline einbinden . . . . .	5
4.2	Quellcode extern einbinden . . . . .	5
4.3	Text, der nicht interpretiert werden soll . . . . .	6
<b>5</b>	<b>Listen</b>	<b>6</b>
5.1	Aufzählungen . . . . .	6
5.1.1	Aufzählungen mit Nummerierung . . . . .	6
5.1.2	Aufzählungen mit anderem Zählzeichen . . . . .	7
5.2	Stichpunkte mit Bullets . . . . .	7
5.3	Auflistung ohne Zeichen . . . . .	7

# 1 Text schreiben

Um einen Absatz zu erzeugen, betätigt man *Enter/Return* zweimal, sodass im Quelltext eine Leerzeile entsteht. Vielfaches benutzen von *Enter/Return* bewirkt nichts. Man kann auch `\` benutzen, allerdings ist davon abzuraten, da der Doppelbackslash für andere Zwecke gedacht ist.

Der Text wird in L<sup>A</sup>T<sub>E</sub>X optimal gesetzt, d.h. wenig Worttrennungen, kleine Wortzwischenräume, Vermeidung mehrerer Leerzeichen untereinander, etc. Darum sollte man so wenig wie möglich manuell dem entgegen wirken. Mehrfaches Anwenden der *Leertaste/Space* bewirkt nichts. Dahingegen kann man seinen Code mit *Returns, Spaces und Tabs* schön leserlich formatieren.

Gedankenstriche – wie diese hier – werden durch `--` erzeugt und sind nicht mit dem Bindestrich `-` zu verwechseln. Auch negative Zahlen werden nicht `-3` sondern im Mathematikmodus `−3` geschrieben.

Eine Textgliederung wird mit `\section{}`, `\subsection{}` und `\subsubsection{}` vorgenommen. So erscheinen diese dann auch im Inhaltsverzeichnis. Sollen manche Überschriften dort nicht erscheinen und keine Nummerierung erhalten, schreibt man `\section*{}`.

# 2 Tabulatoren

Tabellen stehen in einem `\begin{tabular} - \end{tabular}` - Block.

Spalten werden mittels `&` voneinander getrennt und neue Zeilen werden mittels `\` umgebrochen. Auch der Inhalt einer Zeile – damit er nicht zu lang wird – kann manuell mittels `\` umgebrochen werden.

## 2.1 Tabulatoren zur Textstrukturierung

Hier ein einfaches Beispiel:

```
\begin{tabular}[c]{l c r}
  Feld 1.1 & Feld 1.2 & Feld 1.3\\
  Feld 2.1 & Feld 2.2 & Feld 2.3\\
  Feld 3.1 & Feld 3.2 & Feld 3.3\\
\end{tabular}
```

Das sieht dann so aus:

Feld 1.1	Feld 1.2	Feld 1.3
Feld 2.1	Feld 2.2	Feld 2.3
Feld 3.1	Feld 3.2	Feld 3.3

Der Ausdruck `\begin{tabular}[Pos]{Spaltendefinitionen}` erlaubt zusätzliche Einstellungen: `[Pos]` entscheidet über die vertikale Ausrichtung zum Text und kann `c` (*center*), `t` (*on top*) und `b` (*on bottom*) annehmen. In den `{Spaltendefinitionen}` steht für jede Spalte einer der folgenden Buchstaben: `l` (*left*), `r` (*right*) oder `c` (*center*), welche sich auf die horizontale Ausrichtung in den Spalten auswirken.

## 2.2 Tabellen mittels Tabulatoren

Um Tabellen zu zeichnen, benötigt man Linien. Der Befehl `\hline` zwischen den Zeilen erzeugt eine horizontale Linie. Die vertikalen Linien werden in den *Spaltendefinitionen* festgelegt: das Zeichen `|` zwischen den Definitionen sorgt für eine vertikale Linie. Beispiel: `{|l|c|r|}`

Überschrift 1	Überschrift 2	Überschrift 3
linksbündige Spalte	zentriert	rechtsbündige Spalte
Zellen können einfach leer gelassen werden		Feld 2.3
Feld 3.1	Feld 3.2 enthält ein <code>&amp;</code>	Feld 3.3

Ein Umbruch zu langer Zellen ist hier leider nicht möglich. Dazu muss man Tabellen mit fester Spaltenbreite wählen. Allerdings ist der Inhalt dann linksbündig.

1. Zeile:	Es findet ein automatischer Umbruch statt.	kein Kommentar
2. Zeile	Man kann aber auch mittels <code>\newline</code> manuell umbrechen.	kein Kommentar

## 3 Formeln

### 3.1 Formeln im Text

Formeln werden im Text mit `$` eingeleitet und beendet.

So können wir Formeln wie  $\sum_{i=0}^n i = \frac{n(n+1)}{2}$  schreiben.

Wichtige Befehle:

- `\cdot` für die Multiplikation  $a \cdot b$  anstelle von  $a * b$
- `\frac{}{}` für Brüche  $\frac{x+1}{x}$ ,
- `_{}`  für Tiefstellen  $x_0, x_1, \dots, x_n$  und
- `^{}`  für Hochstellen  $e^{-\phi \cdot 2\pi}$ .

### 3.2 Formeln im abgesetzten Mathematikmodus

Komplette Formelabsätze werden in `$$ ... $$` geschrieben. Allerdings ist das Ausrichten der Zeilen zueinander etwas tricky. Für eine schöne Formatierung wird *tabular* benutzt.

$$f(0) = 1$$
$$f(x) = x \cdot f(x - 1)$$

### 3.2.1 Gleichungen

Mit einem `\begin{align} - \end{align}` - Block kann man einen gesamten Absatz für mathematische Formeln erstellt. Dabei wird der Absatz automatisch zentriert, erhält eine Zeilennummer und die Zeilen werden untereinander am Gleichheitszeichen ausgerichtet.

I.S.  $k \rightarrow k + 1$

$$\Leftrightarrow \sum_{i=0}^{k+1} i = \frac{(k+1)(k+2)}{2} \quad (1)$$

$$\Leftrightarrow \sum_{i=0}^k i + (k+1) = \frac{(k+1)(k+2)}{2} \quad (2)$$

$$\Leftrightarrow \text{I.V. } \frac{(k+1)(k+2)}{2} + (k+1) = \frac{(k+1)(k+2)}{2} \quad (3)$$

$$\Leftrightarrow \frac{(k+1)(k+2)}{2} = \frac{(k+1)(k+2)}{2} \quad (4)$$

## 4 Quellcode

### 4.1 Quellcode inline einbinden

Quellcode wird in einem `\begin{lstlisting} - \end{lstlisting}` - Block geschrieben. Einrücken von Zeilen muss mittels Spaces (Leerzeichen) - nicht aber mittels Tabs - erfolgen. Hierbei spielt die Sprache keine Rolle.

```
1  --Pattern-Matching und Guards
2  fac :: Integer->Integer
3  fac 0 = 1          --Rekursionsanker
4  fac n
5      | n>0  = n * fac (n-1)
6      | otherwise = error "Die Eingabe muss eine natürliche Zahl sein."
```

### 4.2 Quellcode extern einbinden

Viel leichter ist es, externe Dateien einzubinden. Sie sollten in einem Verzeichnis des selben Ordners wie die Tex-Datei liegen. Das Einbinden erfolgt über:

`\lstinputlisting[Eigenschaften]{pfad/datei.typ}`. Mit den Eigenschaften werden Sprache, Beschreibungstext und dessen Position und das Label für die Referenz übergeben.

- caption: Beschreibungstext
- label: Label für Referenzen auf das Listing mit `\ref{lst:xyz}`
- captionpos: Position des Beschreibungstextes: t (*top*) und b (*bottom*)
- language: Angabe einer Programmiersprache (z.B. JAVA, XML, C, C++)

Der Vorteil hier: es findet Syntaxhighlighting statt und die Zeilen werden nummeriert.

Beispiel:

```

\lstinputlisting
  [caption={Pattern-Matching und Guards in Haskell}
  \label{lst:haskell},
  captionpos=t,language=HASKELL]
{sources/fac.hs}

```

Und so sieht es dann aus:

Listing 1: Pattern-Matching und Guards in Haskell

```

1 fac :: Integer->Integer
2 fac 0 = 1      --Rekursionsanker
3 fac n
4   | n>0  = n * fac (n-1)
5   | otherwise = error "Die Eingabe muss eine natürliche Zahl sein."

```

### 4.3 Text, der nicht interpretiert werden soll

Manchmal soll ein Teil eines Textes nicht vom  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ -Compiler interpretiert werden. So kann man den Befehl mittels `\verb|<nicht zu interpretierender Code>|` auszeichnen.

Soll dies für einen gesamten Textabschnitt geschehen, geht dies in einem `\begin{verbatim}` - `\end{verbatim}` - Block. Allerdings ist dieser nicht für Quellcode gedacht.

Beispiel:

```

\begin{verbatim}
  Hier werden keine \befehle{} interpretiert. Man kann \\ also alles schreiben: § $ % &
\end {verbatim} % allerdings ohne Leerzeichen

```

## 5 Listen

### 5.1 Aufzählungen

#### 5.1.1 Aufzählungen mit Nummerierung

1. Dies ist eine Aussage.  
    Mit `\\` kann man umbrechen.  
    Und die Umbrüche werden automatisch eingerückt.
2. Hier ist eine weitere Aussage.
  - a) Aufzählungen können verschachtelt sein.
  - b) Wie man gut sehen kann
3. Letzte Aussage.

### **5.1.2 Aufzählungen mit anderem Zählzeichen**

- (i) Analysis I
- (ii) Lineare Algebra I
- (iii) Analysis II
- (iv) Lineare Algebra II
- (v) Analysis III

### **5.2 Stichpunkte mit Bullets**

- Stichpunkt 1
- Stichpunkt 2
- Stichpunkt 3

### **5.3 Auflistung ohne Zeichen**

Zeile 1  
Zeile 2  
Zeile 3