Informatik-Hausaufgaben mit LaTeX setzen

Rocco Rutte <pdmef@cs.tu-berlin.de>

Version 0.1c, 2005/06/09

Dieses Dokument soll das LaTeX-Package tub-info für StudentInnen der TU Berlin dokumentieren, mit denen man ohne viel Handarbeit Hausaufgaben für die Abgabe im Fachbereicht Informatik in hoher Qualität setzen lassen kann.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

Über dieses Dokument

Dieses Dokument soll es als Einstiegshilfe für StudentInnen der TU Berlin aufzeigen, wie man einfach in LaTeX die Informatik-Hausaufgaben für die Abgabe mit tub-info.sty setzen lassen kann. Auch auf Listings von Opal- und Java™-Quelltexten wird eingegangen und richtet sich damit an alle StudentInnen, die Hausaufgaben entweder schriftlich oder elektronisch einreichen müssen.[1]
Dieses Dokument darf selbstverständlich frei kopiert und verändert werden; bei Veränderungen muss lediglich in irgendeiner Form auf den Ursprung hingewiesen werden.
Dieses Dokument ist als PDF-Dokument mit LaTeX-Quelltexten, tub-info.sty sowie Opal- und Java™ Quelltexten verfügbar unter[1].

1 Einführung

1.1 Was ist LaTeX?

TeX ist ein sehr mächtiges Textsatzsystem, das von Donald E. Knuth entwickelt wurde. Es ist *keine* Textverarbeitung und gehört damit nicht in die Kategorie der verbreiteten Office-Pakete–Textsatz umfasst mehr als nur „etwas“ Optik, weil LaTeX eine integrierte *Makrosprache* bietet, die nahezu jede Anpassung quasi durch „Programmierung“ zulässt.
LaTeX (Aussprache: 'la:teç, „latech“ mit weichem „ch“) ist eine Sammlung nützlicher Makros für TeX, die die Arbeit erheblich vereinfachen.
Entscheidender Bestandteil des Konzeptes dabei ist, dass Autoren in LaTeX nur die *logische Struktur* des Textes vorgeben, also zum Beispiel nicht vorgeben, *wie* eine Überschrift auszusehen hat (Schrift, Ausrichtung, etc) sondern nur, *dass* es eine Überschrift ist. Um das Layout kümmert sich LaTeX selbst, weil es ja nur von der Struktur abhängt.[2]
Auf Makros und „reinem“ (engl.: *plain*) TeX aufbauend, gibt es sehr viele fertige Pakete (sog. *Packages*) und Dokumentklassen, mit denen man vom professionellen Brief (mit Markierungen strickt nach DIN-Norm), über Diplomarbeiten bis hin zu Büchern und Noten praktisch alles in hoher Qualität setzen (lassen) kann.
Mehr Informationen zu LaTeX gibt es im Internet bei Wikipedia[2], Dante e.V.[3] und CTAN[4].

1.2 Warum LaTeX nutzen?

Mit LaTeX kann man qualitativ hochwertige Dokumente erstellen, weil durch die eingebaute Makrosprache sehr große Teile des Layouts automatisch erstellt und im ganzen Dokument konstistent gehalten werden (können).
Dadurch wird dem Autor ein sehr großer Teil der kleinen, aber wichtigen Details abgenommen: stets konsistente Nummerierung von Verzeichnissen, Abbildungen, Tabellen, Fußnoten, ..., Abschnitten, automatisch (fast immer) richtige Silbentrennung je nach

[1] da man mit LaTeX auch qualitativ hochwertige PDF-/PostScript-Dokumente erzeugen kann
[2] konzeptionell ähnlich dazu ist zum Beispiel HTML oder XML, was auch „nur“ Auszeichnungssprachen (engl.: *Markup Language*) sind

verwendeter Sprache, konsistente und richtige Abstände, vollautomatische Verzeichnisse und Indizes usw. Diese Details sind gerade in wissenschaftlich-orientierten Dokumenten unerlässlich, vor allem, wenn man mit dem eigenen Namen für etwas „gerade stehen“ muss. Die Qualität der zur Arbeit verwendeten Werkzeuge sagt viel über den Autor aus. Eine der großen Stärken von LaTeX sind die fest eingebauten Möglichkeiten zum Formelsatz. Hat man die Grundlagen gelernt, ist es nicht schwer

$$\underbrace{\ln\left(\overbrace{\lim_{\delta\to\infty}\left(\underbrace{|(X')^{-1}-(X^{-1})'|}_{1}!+\frac{1}{\delta}\right)}^{e}{}^{\delta}\right)}_{1}+\left(\sin^2 q+\cos^2 q\right)=\sum_{n=0}^{\infty}\frac{\cosh p\sqrt{1-\tanh^2 p}}{2^n}$$

oder auch „nur“

$$F_n=\begin{cases}1 & \text{für } n=0\vee n=1\\ F_{n-2}+F_{n-1} & \text{sonst}\end{cases}$$

einfach in einem Dokument zu verwenden, wenn man es benötigt.
LaTeX-Dokumente sind reiner (ASCII-)Text, was eine Menge Vorteile hat: man braucht nur den einfachsten Texteditor, um Dokumente zu schreiben, man kann sie bequem mit Versionskontrollsystemen wie CVS oder Subversion verwalten, mehrere Autoren können problemlos gleichzeitig an verschiedenen Teilen arbeiten, ein Dokument bleibt bis in alle Ewigkeit lesbar, selbst wenn es irgendwann LaTeX nicht mehr geben sollte, ...
Diese Eigenschaften erweisen sich allerdings erst im „alltäglichen“ Gebrauch von LaTeX als überaus nützlich und hilfreich–nur für das Setzen von Hausaufgaben spielen sie eine untergeordnete Rolle.
Warum also das ganze?
Wie die Minimalbeispiele zeigen werden, muss man *einmal* ein wenig Zeit in LaTeX investieren, um schnell zu qualitativ hochwertigen Ergebnissen zu kommen, ... was widerum Motivation genug für eine intensivere Beschäftigung mit LaTeX sein soll. Dies ist für alle Arten von veröffentlichten Dokumenten (Vortragsunterlagen, Abschlussarbeiten, Papers) später ohnehin notwendig, weil es kaum vergleichbar leistungsfähig und flexible Textsysteme gibt.

1.3 Motivation: (prominente) LaTeX-Beispiele

Gerade die leistungsfähigen Möglichkeiten für den Formelsatz machen LaTeX für Mathematiker und Ingeneure interessant: so sind Skripte an der TU[3] sowie Übungs- und Klausurblätter so geschrieben. Auch in Nicht-Mathematik Fachrichtungen (zum Beispiel in der Informatik) wird LaTeX dafür verwendet. Und weil es in diesem Dokument auch

[3] Analysis I/II für Ing., Lineare Algebra für Ing., ...

um das Setzen von Opal-Quelltexten gehen wird: auch das Buch über funktionale Programmierung von Prof. Pepper ist in LaTeX geschrieben.
Außerhalb der TU finden sich auch zahlreiche weitere Bücher,[4] Papers und Vortragsfolien, die mit LaTeX erstellt wurden.

1.4 LaTeX im CS-Netz

Auf den Rechnern im CS-Netz ist teTeX[5] installiert (s. Abschnitt 1.5). Um es nutzen zu können, reicht es, den Pfad `/usr/TeX/teTeX-3.0/bin` in die Konfigurationsdatei der Shell einzutragen.[5]
Je nachdem, welchen Typ Ausgabe man gern hätte (DVI, PostScript, PDF), unterscheidet sich die Erstellung der Dokumente. Hat man eine Datei `bsp1.tex`, dann kann man:

- mit `latex bsp1.tex` eine DVI-Datei `bsp1.dvi` erzeugen
- mit `dvips bsp1.dvi -o bsp1.ps` daraus dann eine PostScript-Datei erzeugen
- oder mit `pdflatex bsp1.tex` die PDF-Datei `bsp1.pdf` erzeugen

1.5 Bezugsquellen

Aus Sicht des Nutzers gibt es leider nicht *das* TeX sondern verschiedene TeX-*Distributionen*:

teTeX : Es wird im CS-Netz benutzt und ist unter Unix-artigen Systemen wie Solaris (CS-Netz), Linux-Distributionen und den BSD-Derivaten der Quasi-Standard. Will man teTeX selbst installieren, hängt es von dem verwendeten System bzw. von der Paketverwaltung ab, wie man es installieren muss. Unter Windows enthält die Cygwin-Umgebung[8] ebenfalls alle benötigten Pakete. Mehr Informationen zu teTeX gibt es unter [5].

MiKTeX Im Zusammenhang mit Windows und LaTeX wird auch gern MiKTeX[6] empfohlen.

TeXLive Dieses Projekt bietet eine fertige „Live-CD“ für LaTeX an, mit der man ohne Installation auf zahlreichen Plattformen (diverse Vendor Unices, Linux, BSD, Windows, MacOS X) direkt loslegen kann—aber auch Installationen sind möglich. Mehr Informationen gibt es unter [9].

Je nachdem, welches TeX auf welchem Betriebsystem zusammen mit welchem Editor verwendet wird, empfehlen sich zusätzliche Hilfen wie die LaTeX-Suite[12] für vim, AUCTeX und Preview-LaTeX[13] für (x)emacs oder grafische Frontends wie LyX[14], die jeweils alle auf unterschiedlichen Plattformen meist frei verfügbar sind.[6]

[4] Knuth: The Art of Computer Programming; Bronstein: Taschenbuch der Mathematik; Formeln und Hilfen zur Höheren Mathematik, ...
[5] z.B., indem `export PATH="$PATH:/usr/TeX/teTeX-3.0/bin"` in die Datei `.bashrc` eingetragen wird
[6] „frei“ im Sinne von „Freibier“ und „gratis“

1.6 Installation der LaTeX-Packages

Folgende Schritte sind nötig, um tub-info.sty zu installieren:

1. mit `mkdir -p ~/texmf/tex/latex/tub` das Verzeichnis `~/texmf/tex/latex/tub` anlegen
2. die Datei tub-info.sty von <`http://user.cs.tu-berlin.de/~pdmef/opal/`> in dieses Verzeichnis kopieren
3. `texhash ~/texmf` aufrufen

2 Lösungen abgefertig setzen

2.1 Quelltexte setzen

2.1.1 Minimal-Beispiel

Ein Minimal-Beispiel ist:

```
\documentclass[a4paper]{scrartcl}
\usepackage{tub-info}
\begin{document}
  \opalcode{Test1.sign}
  \opalcode{Test1.impl}
  \javacode{Code1.java}
\end{document}
```

Listing 1: Minimal-Beispiel für Code-Listings

Für all die, denen TeX neu ist, hier ein paar kurzer Erklärungen:

- `\documentclass[a4paper]{scrartcl}` wählt die Dokumentklasse scrartcl mit der Option `a4paper` aus (damit u.a. Seitenränder für unsere Augen verträglich sind)
- `\usepackage{tub-info}` lädt tub-info.sty
- `\begin{document}` ist der Beginn des Inhalts des Dokuments
- `\opalcode{Test1.sign}`, `\opalcode{Test1.impl}` und `\javacode{Code1.java}` ist in tub-info.sty definiert und ruft Funktionen aus dem listings-Package auf, die den Quellcode mit Syntax-Highlighting, Rand, Zeilennummern und Beschriftung setzen.
- `\end{document}` schließt das Dokument ab

Heißt diese Datei zum Beispiel test1.tex, dann rutft man einfach pdflatex test1.tex auf und erhält ein PDF-Dokument, s. Abbildung 1 auf der nächsten Seite.

Listing 1: Test1.sign

```
SIGNATURE Test1

IMPORT Real ONLY real

-- shape : ℝ≥0 ↦ ℝ≥0
-- shape (r) = π · r^2
FUN shape : real -> real
```

Listing 2: Test1.impl

```
IMPLEMENTATION Test1

IMPORT Real COMPLETELY

DEF shape (r) == pi * (r^2)
```

Listing 3: Code1.java

```
// does arr_i ⇔ arr_j ...
private void swap1 (int[] arr, int i, int j) {
  // ... without a temporary variable
  // PRE: arr_i = N ∧ arr_j = M
  // arr_j = M ∧ (arr_i + arr_j) − arr_j = N ⇔ arr_j = M ∧ arr_i = N
  arr[i] = arr[i] + arr[j];
  // arr_i − (arr_i − arr_j) = M ∧ arr_i − arr_j = N ⇔ arr_j = M ∧ arr_i − arr_j = N
  arr[j] = arr[i] - arr[j];
  // arr_i − arr_j = M ∧ arr_j = N
  arr[i] = arr[i] - arr[j];
  // POST: arr_i = M ∧ arr_j = N
}
```

1

Abbildung 1: Minimal-Beispiel für Code-Listings

2.1.2 Weitere Möglichkeiten

Wie man in Abbildung 1 auf der vorherigen Seite schon sieht, kann man auch Formeln benutzen—doch bevor ich erkläre, wie das geht, noch ein paar andere Erklärungen.

Titel vorgeben

Für den Fall, dass die Quelltexte nicht im gleichen Verzeichnis wie die LaTeX-Quellen sind, muss man einen Pfad für die Quellen angeben[7]. Dann ist es unschön, wenn der Pfad auch im Titel des Listings steht. Die Befehle **\opalcode** und **\javacode** verstehen nicht nur ein „Pflicht"- sondern auch ein optionales Argument: Pflicht ist der Pfad zur Quelltext-Datei und optional der Titel des Listings. In LaTeX sieht das so aus:

```
\documentclass[a4paper]{scrartcl}
\usepackage{tub-info}
\begin{document}
  \opalcode[Test1.sign]{/home/p/pdmef/opal/Test1.sign}
  \opalcode[Test1.impl]{/home/p/pdmef/opal/Test1.impl}
  \javacode[Code1.java]{/home/p/pdmef/java/Code1.java}
\end{document}
```

Listing 2: Optionale Titel von Code-Listings

Es wird die Datei mit vollem Pfad eingebunden, der Titel ist jedoch nur Test1.sign, Test1.impl bzw. Code1.java. Der Titel muss natürlich nicht dem Dateinamen entsprechen sondern ist frei wählbar.

Formeln & Co.

In LaTeX kann man ohne weitere Packages zwar problemlos auch Quelltexte setzen, jedoch komplett ohne Syntax-Highlighting und Nachbearbeitung. Da Opal sehr stark an die Ausdrucksweise der Mathemaik angelehnt ist und man für Java™-Quellen das Hoare-Kalkül praktischerweise sehr mathematisch formuliert, kann es hilfreich sein, wenn man auch solche formale Definitionen oder Beschreibungen in Form von Formeln als Autor nutzen könnte.
Nutzt man das listings-Package zum Setzen von Quelltexten, kann man sehr flexibel festlegen, wie was wann von LaTeX doch interpretiert werden soll. Ich habe mich für Kommentare entschieden, die *stets komplett* von LaTeX bearbeitet werden. Darin funktionieren dann Verweise, Formeln, etc.
In LaTeX sieht die formale Definition für eine Abbildung, die die Kreisfläche S eines Kreises mit Radius r liefert, so aus:

```
-- $\mbox{shape}\, : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$
-- $\mbox{shape}\, (r) = \pi \cdot r^2$
```

Diese formale Definition schreibt man einfach als Kommentar in den Opal-Quelltext und

[7] die elegante Shell-basierte Lösung ist natürlich der Weg über `$TEXINPUTS`

der **\opalcode**-Befehl lässt es dann umsetzen. Das Ergebnis sieht man in Abbildung 1 auf Seite 7.
Analog gilt dies natürlich für JAVA™-Code:

```
// this does $arr_i \Leftrightarrow arr_j$ \ldots
```

2.1.3 Syntax-Hervorhebungen

Weil im verwendeten listings-Package eine Sprachdefinition für OPAL nicht vorhanden ist, habe ich sie selbst geschrieben: Tabelle 1 listet Details zu den OPAL-Sprachelementen, die durch diese Sprachdefinition in hervorgehoben werden.

Sprachelement	Beispiel	Stil
normaler Code	—	`txtt`
Kommentare	—	Dokument-Standard
Schlüsselworte	**IF**, **THEN**, **ELSE**	fett
Datentypen	nur **bool** und **denotation**	fett
Werte von **bool** und **denotation**	*true*, *"3.14"*!	kursiv
Symbole	nur **->**, == und **	fett

Tabelle 1: OPAL Syntax-Hervorhebungen

2.2 Fertige Abgaben mit setzen

Für Einsteiger in LaTeX & Co. ist es vielleicht zwar hilfreich, Code-Listings ansprechend setzen zu können. . . aber der Dokumentkopf fehlt noch.

2.2.1 Minimal-Beispiel

Auch dazu ein Minimal-Beispiel:

```
\documentclass[a4paper]{scrartcl}
\newcommand{\gruppe}[0]{Gruppe 0}
\newcommand{\lv}[0]{Informatik 1}
\usepackage{tub-info}
\begin{document}

  \blatt{2}{15.--19.11.2004}

    \aufgabe{}

      Das Programm \textsf{WhatsThat} liefert die Quadratwurzel.
```

```
    \aufgabe{Einfache Geomtrie in Opal}

      \opalcode{Test2.sign}
      \opalcode{Test2.impl}

\end{document}
```

Listing 3: Minimal-Beispiel für fertige Abgaben

Nach einem LaTeX-Lauf erhält man eine abgabefertige Datei, siehe Abbildung 2 auf der nächsten Seite.
Und auch dazu ein paar Erklärungen:

- Mit **\newcommand** kann man in LaTeX neue Befehle definieren: zwischen die geschweiften Klammern in den Definitionen von **\gruppe** und **\lv** schreibt man einach die Nummer der Gruppe sowie den Nahmen der Lehrveranstaltung bevor man tub-info.sty lädt. Auch wenn das kompliziert scheint, hat es doch den Vorteil, dass man die Datei tub-info.sty für alle schriftlichen Abgaben im Fachbereich Informatik bzw. nur eine LaTeX-Datei für das ganze Semester nutzen kann.
- Der Befehl **\blatt** erwartet als Argumente die laufende Nummer des Aufgabenblattes sowie das Abgabdedatum. Außerdem wird damit auch gleich der Dokumentkopf mit Rahmen und Überschrift erzeugt.
- Mit dem Befehl **\aufgabe** kann man, wie der Name vermuten lässt, den Text „Aufgabe“ erzeugen lassen. Alle Aufgaben, die zu einem Blatt gehören, werden natürlich richtig nummeriert. Wird in den geschweiften Klammern außerdem noch ein Beschreibungstext angegeben, steht der dann natürlich auch auf dem Blatt.

2.2.2 Schriftarten

Bis auf `txtt` aus dem txfonts-Package als Fixbreitenschrift bleiben alle Schriftarten unverändert. Nutzt man beispielsweise eine Klasse aus Koma-Script[11] wie scrartcl statt article, dann werden Überschriften entsprechend an den Dokument-Standard angepasst.

2.2.3 Weitere Möglichkeiten

Mehrere Blätter pro LaTeX-Dokument

Man kann natürlich die Lösungen eines Semesters in eine Datei schreiben und einfach mehrere **\blatt**-Befehle nutzen. Die Seitenzahlen sowie Nummerierung von Aufgaben und Listings beginnen stets von neuem.
Aber Vorsicht: tub-info.sty trickst etwas, um Nummerierungen stets von neuem beginnen zu lassen bzw. setzt Zähler zurück, die man vielleicht besser nicht zurücksetzen sollte. Deshalb kann und wird es bei Verweisen quer durch mehrere Lösungsblätter zu Problemen kommen.

Technische Universität Berlin Lösungen Gruppe 0	Informatik 1 2. Aufgabenblatt Abgabe: 15.–19.11.2004

Hausaufgaben

1. Aufgabe
Das Programm WhatsThat liefert die Quadratwurzel.

2. Aufgabe: Einfache Geomtrie in Opal

Listing 1: Test2.sign

```
SIGNATURE Test2

IMPORT Real ONLY real

-- shape : ℝ≥0 ↦ ℝ≥0
-- shape (r) = π · r^2
FUN shape : real -> real

-- vol : ℝ≥0 × ℝ≥0 ↦ ℝ≥0
-- vol (r, h) = shape (r) · h
FUN vol : real ** real -> real
```

Listing 2: Test2.impl

```
IMPLEMENTATION Test2

IMPORT Real COMPLETELY

DEF shape (r) == pi * (r^2)

DEF vol (r, h) == shape (r) * h
```


Abbildung 2: Einfaches abgabefertiges PDF-Dokument

PDF-Bookmarks
Wenn mit pdflatex PDF-Dateien erzeugt werden, werden im Inhaltsverzeichnis automatisch sog. *Bookmarks* erstellt: diese beinhalten sowohl die einzelnen Blätter als auch alle Aufgaben, so dass man schnell in größeren Dokumenten zum gewünschten Ziel springen, bestimmte Teile auf und zu klappen kann, etc.

Sonstige Formatierungen
Die Datei tub-info.sty enthält noch eine Vorgabe für das optisch ansprechende Setzen von Aufgaben mit Teillösungen: mit `\begin{teilaufgaben}` bzw. `\end{teilaufgaben}` gibt man diese Umgebung an. Dazwischen kann man für jede Teilaufgabe `\teilaufgabe` nutzen.

2.3 Bugs

Derzeit sind keine Fehler bekannt; falls doch welche auftauchen, dann bitte ich um eine E-Mail an <pdmef@cs.tu-berlin.de>.

3 Erweiterte Möglichkeiten

3.1 Anpassungen allgemein

Listings: allgemein Für Anpassungen, die das Aussehen des Listings beeinflussen sollen (z.B. den Stil für Kommentare ändern, die Nummerierung abschalten, etc), sollte man die Dokumentation des listings-Packages[10] konsultieren und tub-info.sty entsprechend anpassen.

Listings: Schriftart Leider gibt es die Standard-Typewriter-Schriftart von LaTeX nicht in einer fetten Variante, zumindest nicht ohne weitere LaTeX-Kenntnisse bzw. nicht, wenn es noch schön aussehen soll. Daher wird zum Setzen von Listings `txtt` aus dem txfonts-Package genutzt. Dies kann in tub-info.sty geändert werden.

Listings: Syntax-Highlighting Da in Opal praktisch keine Datentypen außer **`bool`** und **`denotation`** fest vordefiniert sind,[8] werden auch nur diese vom Syntax-Highlighting erkannt. Wer das ändern möchte, muss Hand an tub-info.sty legen.

Seitenränder Verwendet man die Dokumentklasse article sind Seitenränder mitunter recht breit, so dass man nur Quelltexte mit relativ kurzen Zeilen setzen kann. Hierfür gibt es mehrere Möglichkeiten zur Abhilfe:

1. statt der Dokumentklasse article die Klasse scrartcl aus Koma-Skript[9][11] mit der Option `a4paper` verwenden

[8] alle anderen wie `real` zum Beispiel sind selbst in Opal programmiert

[9] Koma-Skript ist eine Sammlung von Klassen, die die LaTeX-Standards verändern und flexiblere Möglichkeiten zur Anpassung bieten; sich mit den Koma-Skript-Klassen zu beschäftigen kann auch sinnvoll sein, wenn man langfristig (auch größere) Projekte mit LaTeX bearbeiten will

2. gesamte Schriftgröße des Dokuments herabsetzen (zum Beispiel mit `\documentclass[9pt,...]{...}`)
3. Standard-Schriftgröße nur für Listings herabsetzen (dafür muss die `basicstyle`-Option in tub-info.sty angepasst werden)
4. Quellcode besser strukturieren, kürzere Zeilen und damit lesbaren Code schreiben
5. ...

3.2 Interface zu LaTeX

In tub-info.sty sind nur ein paar kurze Befehle definiert, die man häufiger benötigt, um Hausaufgaben setzen zu können.
Daneben steht natürlich die volle Bandbreit an Möglichkeiten zur Verfügung, von Fußnoten und Verweisen innerhalb von Quelltextlistings bis hin zu automatischen Indizes und Verzeichnissen, eingebundenen Abbildungen, etc.

3.3 Interface zum listings-Package

3.3.1 Listingstil(e)

Mit dem listings-Package kann man sich zur Vereinfachung Stile definieren: tub-info.sty definiert die Stile `opal` und `java`, die man direkt für Befehle des listings-Packages nutzen kann. Dafür muss man nur tub-info.sty benutzen und `style=opal` bzw. `style=java` als Parameter in den Listing-Befehlen setzen.
Beispielsweise setzt man ein Listing der Zeilen 22 bis 24 von WhatsThat.impl, in dem jede Zeile nummeriert und nur ein linker Rand gesetzt wird, mit dem Befehl:

```
\lstinputlisting[style=opal,frame=l,stepnumber=1,
  firstline=22,lastline=24]{WhatsThat.impl}
```

Das sieht dann so aus:

```
22 -- testet, ob x ≈ y
23 FUN ~~ : real ** real -> bool
24 DEF ~~(x,y) == abs(x-y) < !("0.0000001")
```

Listing 4: Spezielle listings-Funktionen mit dem `opal`-Stil

Selbiges gilt für JAVA™-Listings mit dem `java`-Stil.
Erweiterterte Stile mit Literaten ist durch `style=opalext` und `style=javaext` verfügbar, s. Abschnitt 3.3.2.

3.3.2 Literate

Weiterhin bietet das listings-Package das als experimentell markierte Feature „literate programming“. Damit kann man angeben, welche Textsequenzen in Quelltexten wie ersetzt werden sollen.
Der OPAL-Quelltext:

```
FUN foo : alpha ** nat -> nat -> seq[alpha]
DEF foo (bar, x0) == \\x1.bar :: <>
```

wird damit zu:

```
FUN foo : α × nat ↦ nat ↦ seq[α]
DEF foo (bar, x0) == λ x1.bar :: ∅
```

Listing 5: Literate für OPAL-Code

Der JAVA™-Quelltext:

```
while (i <= n) {
  if (i != 0) {
  }
}
```

wird damit zu:

```
while (i ≤ n) {
  if (i ≠ 0) {
  }
}
```

Listing 6: Literate für JAVA™-Code

Leider ist die Unterstützung dafür nicht sehr ausgereift,[10] weshalb ich mich entschieden habe, nicht sehr viele Symbole anzugeben, siehe Tabellen 2 und 3 auf der nächsten Seite, und diese Ersetzungen nicht zum Normalverhalten zu machen. Es ist eher ein „proof of concept“, was man mit LaTeX alles anstellen kann.
Tauscht man `\usepackage{tub-info}` mit `\usepackage[symbols]{tub-info}`, sind diese automatisch aktiv.

normal	literate	normal	literate
`->`	$\mapsto$	`**`	$\times$
`<>`	$\emptyset$	`\\`	λ
`alpha`	α	`beta`	β
`Pi`	π		

Tabelle 2: Literate für OPAL-Quelltexte

[10] in OPAL-Quelltext, zum Beispiel, reicht schon das Wort „Beispiel“ in einem Kommentar, um aus dem Teil „pi“ ein „π“ zu machen...

normal	literate	normal	literate
!=	$\neq$	>=	$\geq$
<=	$\leq$		

Tabelle 3: Literate für JAVA™-Quelltexte

Literatur

[1] <http://user.cs.tu-berlin.de/~pdmef/opal/>.

[2] Eintrag zu „TeX“/„LaTeX“ bei Wikipedia.
<http://de.wikipedia.org/wiki/TeX> und
<http://de.wikipedia.org/wiki/LaTeX>

[3] Dante e.V.
<http://www.dante.de/tex/>

[4] CTAN, Comprehensive TEX Archive Network.
<http://www.ctan.org/>

[5] teTEX: TEX-Distribution für Unices.
<http://www.tug.org/tetex/>

[6] MiKTEX: TEX-Distribution für Windows.
<http://www.miktex.org/>

[7] WinShell: grafisches TEX-Frontend für Windows.
<http://www.winshell.de/>

[8] Cygwin: Unix-artige Umgebung für Windows.
<http://www.cygwin.com/>

[9] TEX Live: TEX-Live-CD für zahlreiche Betriebssysteme.
<http://www.tug.org/texlive/>

[10] listings.sty: Package zum optisch ansprechenden Setzen von Quelltexten aller Art. Bezug „offiziell“ bei CTAN[4]:
<http://www.ctan.org/tex-archive/macros/latex/contrib/listings>
oder fertig präpariert von
<http://user.cs.tu-berlin.de/~pdmef/listings/>

[11] Koma-Skript: <http://komascript.de/>

[12] LATEX Suite für vim: <http://vim-latex.sourceforgen.et>

[13] AUCTEX: <http://www.gnu.org/software/auctex/> und Preview-LATEX: <http://preview-latex.sourceforge.net> für (x)emacs

[14] LyX grafisches WYSIWYG-Frontend für LaTeX: <http://www.lyx.org/>