Einführung in die maschinennahe, imperative, funktionale, relationale und objektorientierte Programmierung

-

EMIFROP 0.28

Markus Alpers
B.Sc. und Ausbilder f. Industriekaufleute

7. April 2016

Inhaltsverzeichnis

Hinweis bezüglich diskriminierender Formulierungen

In diesem Text wurde darauf geachtet Formulierungen zu vermeiden, die diskriminierend verstanden werden können. Im Sinne der Lesbarkeit wurden dabei Formulierungen wie „Informatiker und Informatikerinnen“ durch „InformatikerInnen“ (mit großem i) ersetzt. An anderen Stellen habe ich Formen wie eine/einer durch eineR zusammengefasst. Hier berufe ich mich auf den Artikel „Sprache und Ungleichheit“ der Bundeszentrale für politische Bildung, kurz BpB, vom 16. April 2014, insbesondere auf den Absatz „Zum Umgang mit diskriminierender Sprache“, online abrufbar unter:

`http://www.bpb.de/apuz/130411/sprache-und-ungleichheit?p=all`

Sollten Sie dennoch Formulierungen entdecken, die diesem Anspruch nicht entsprechen, möchte ich Sie bitten, mir eine entsprechende Nachricht zu senden, denn es ist mir wichtig, Ihnen mit diesem Buch eine wertvolle Unterstützung beim Start in die faszinierende Welt der Informatik zu bieten. Das sollte nicht durch verletzte Gefühle in Folge missverständlicher Formulierungen torpediert werden.

Sie erreichen mich unter `markus.alpers@haw-hamburg.de`.

Hinweis zur Lizenz

zenz"´. Wenn Sie einen Teil finden, in dem diese „Hinweise zur Lizenz"´ nicht zu finden ist, dann dürfen Sie für den persönlichen Gebrauch dennoch Kopien davon anfertigen und Sie dürfen diese Kopien außerhalb von kommerziellen Projekten frei verwenden.

Hinweis zur Verwendbarkeit in wissenschaftlichen Arbeiten

Bitte beachten Sie dabei aber, dass die Verwendung dieses Textes im Rahmen wissenschaftlicher Publikationen zurzeit aus anderen Gründen problematisch ist:

- Wie viele andere Quellen, die frei im Internet verfügbar sind, wurde auch dieser Text bislang nicht durch einen nachweislich entsprechend qualifizierten Lektor verifiziert. Damit genügen Zitate aus diesem Band streng genommen noch nicht den Ansprüchen wissenschaftlicher Arbeiten.
- Weiterhin fehlen in diesem Buch häufig die für eine wissenschaftliche Arbeit nötigen Quellenangaben. Wann immer Sie in einer wissenschaftlichen Arbeit eine Behauptung aufstellen, müssen Sie diese durch einen Beleg oder Beweis untermauern. Ein solcher Belege/Beweis kann zum einen eine andere wissenschaftliche Arbeit sind, in der bewiesen wurde, dass die Behauptung den Tatsachen entspricht oder es muss ein eigenständiger Beweis für die Aussage sein.

Bitte beachten Sie außerdem, dass dieses Buch eine Konvention nutzt, die in wissenschaftlichen Arbeiten verpönt ist: Wenn in einer wissenschaftlichen Arbeit ein Begriff hervorgehoben wird, dann wird dazu kursive Schrift verwendet. In diesem Buch verwende ich dagegen Fettdruck, da es vielen Menschen schwer fällt, einen kursiv gedruckten Begriff schnell zu finden und ich mir wünsche, dass Sie es möglichst effizient auch als Nachschlagewerk nutzen können.

Dieses Buch ist allerdings auch **nicht als wissenschaftliche Arbeit**, also als Ergebnis einer Forschung über Programmiersprachen, **sondern als Lehrwerk für den Einstieg in die Programmierung gedacht**. Es dient somit nicht dazu, Ihnen die theoretischen Grundlagen der Programmierung zu vermitteln, sondern dazu, ihnen eine fundierte Unterstützung beim Einstieg in die Programmierung anzubieten.

Teil I

Einfache Einführung in die imperative Programmierung

Kapitel 1

Typische Irrtümer darüber, was Programmieren ist.

Kapitel 2

Von der Nachrichtentechnik zur Programmierung

Kapitel 3

Vorbereitung fürs Programmieren

Kapitel 4

Ausgewählte Programmiersprachen

Kapitel 5

Grundlagen der Entwicklung verteilter Anwendungen

Kapitel 6

Einstieg in HTML 5

Kapitel 7

ML, Teil 2 - Textverarbeitung mit LaTeX

Dieses Kapitel ist nicht Teil des Kurses „Einführung ins Programmieren“ oder von „Programmieren 1“, aber Sie sollten es dennoch durcharbeiten. Zum einen lernen Sie hier eine weitere Markup Language kennen und können dadurch einige Gemeinsamkeiten und Unterschiede erlernen. Zum anderen ist LaTeX die weltweit am häufigsten eingesetzte Textverarbeitung für wissenschaftliche Texte. An einigen Hochschulen wird sogar verlangt, dass Arbeiten als LaTeX-Dokumente eingereicht werden.

LaTeX ist aus Sicht der Programmierung eine Markup Language. Im Gegensatz zu HTML handelt es sich hier aber um eine Sprache, die dafür gedacht ist, verschiedenste (vorrangig gedruckte) Dokumente zu erzeugen. Entwickelt wurde sie von Leslie Lamport, einem der Träger des **Turing Award**. (Wem das kein Begriff ist: Der Turing Award wird als Nobelpreis der Informatik bezeichnet.) Im Gegensatz zu TeX, von dem LaTeX eine Erweiterung ist müssen wir hier nur all das Programmieren, was anders als beim Standard-Dokument ist.

Dieser Kurs ist eine knappe Einführung. Umfangreichere Kurs werden an den meisten Hochschulen aber auch direkt von lokalen LaTeX-Gruppen angeboten. Im Department Medientechnik bietet beispielsweise Prof. Görne alle zwei Semester eine Schulung an, die für die Mitglieder des Departments kostenlos ist. Hier können Sie natürlich auch individuelle Fragen stellen.

LaTeX hat einige essentielle Vorteile gegenüber den meisten Textverarbeitungen, die Sie aus dem Alltag kennen. Das hier ist nur eine Auswahl:

- Genau wie in HTML kümmern Sie sich um den Inhalt und nicht um die Darstellung.

- Sie können aber jedes Detail anpassen.
- Da Sie direkt im Code arbeiten und auch Einrückungen manuell programmieren kann LaTeX Ihre Arbeit nicht torpedieren.
- Es gibt eine große Community, die im Bedarfsfall helfen kann.
- Es ist vollständig kostenlos.
- Formeln lassen sich sehr einfach eintragen und ändern.
- Überschriften werden wissenschaftliche nummeriert.
- Inhaltsverzeichnisse, Glossare, nummerierte Abschnitte und ähnliches passen sich automatisch an Änderungen an.

Wie bei allen Markup Languages gibt es allerdings einen Nachteil:

- Sie müssen sich in die Programmierung einarbeiten. Aber wenn Sie HTML beherrschen, dann wird das für Sie keinen allzu großen Anspruch darstellen. Streckenweise ist es einfacher als die Einarbeitung in HTML5, da wir hier (genauer gesagt in LaTeX 2) kein semantisches Web haben und uns auch nicht um Aspekte des Responsive Design kümmern müssen.

Zur Erinnerung: Installieren Sie bitte `TeXStudio`, um mit der Programmierung zu beginnen.

7.1 Grundlagen

Das meiste, was Sie an Grundlagen wissen müssen kennen Sie jetzt schon: Sie wissen was eine Markup Language ist. Bei LaTeX gibt es eine andere Syntax und andere Bezeichnungen für Elemente und Container, die müssen Sie also lernen. Außerdem gibt es einige zusätzliche Container, mit denen Sie z.B. ein Inhaltsverzeichnis generieren können. Ähnlich wie in HTML5 gibt es aber nur im Bedarfsfall ein schließendes Tag.

7.1.1 Struktur eines LaTeX-Dokuments

In HTML kennen sie die drei zentralen Container html, head und body.

In Latex haben wir keinen html-Container bzw. keinen latex-Container und auch keine Doctype Declaration.

Stattdessen sprechen wir von einer **Präambel**. Diese enthält alles, was wir bei HTML als Attribut im `html`-Tag und im `<head>` untergebracht haben. Danach folgt der eigentliche Inhalt des Dokuments, für den wir keine spezielle Bezeichnung haben.

In LaTeX gibt es leider keinen Bezeichner der dem `Tag` in HTML entspricht. Um es Ihnen beim Einstieg einfacher zu machen werde ich den Begriff in diesem Kapitel dennoch verwenden. Allerdings gibt es hier den Begriff des `Elements`, der für eine kleinste Einheit eines Dokuments steht.

7.1.2 Einfache Präambel

Die folgende Präambel beinhaltet alles, was Sie für die meisten Dokumenten benötigen dürften:

```
\documentclass[11pt, a4paper, oneside, draft]{book}

\usepackage{palatino, url}
\usepackage[ngermanb]{babel}
\usepackage[utf8]{inputenc}

\setlength{\parindent}{0cm}
```

Wie Sie sehen nutzen wir in LaTex keine spitzen Klammern, um Tags anzuzeigen, sondern wir beginnen jedes „Tag“ mit einem Backslash`\`. Bitte beachten Sie den Unterschied: Es handelt sich nicht um den Slash, den Sie mit der Tastenkombination `Shift 7` erhalten, sondern mit der Tastenkombination `Alt gr ß`. Sie müssen dazu also die rechte Alt-Taste (die mit `Alt gr` beschriftet sein sollte) drücken, gedrückt halten und zusätzlich das ß. Das ist anfangs etwas ungewohnt, gibt sich aber mit der Zeit.

Struktur von Umgebungen

- Jeder LaTeX-Container beginnt mit einem Backslash `\` .
- Danach folgt der Bezeichner.
- Es kann ein paar eckiger Klammern mit einem oder mehreren Einträgen folgen. `[]`
- Es können ein oder mehrere Paare geschweifter Klammern mit jeweils einem oder mehreren Einträgen folgen. `{}`

Dabei gibt es keine allzu genaue Systematik dafür, was in den geschweiften oder eckigen Klammern steht. Die geschweiften Klammern entsprechen

aber meist dem Inhalt eines Containers.

documentclass

Die documentclass gibt in geschweiften Klammern an, um welche Art von Dokument es sich handelt. Daraus folgen einige Standardeinstellungen, die Sie aber auch ändern können. Hier eine Auswahl:

- `book` ist dafür gedacht, um die typischen Elemente eines Buches bereit zu stellen: Teile, Kapitel, usw.
- `report` ist für umfangreiche Reportagen gedacht, also auch beispielsweise für Forschungsberichte, die eine Zusammenfassung und mehrer Kapitel umfassen können.
- `article` ist für Artikel gedacht, die z.B. in Zeitschiften erscheinen sollen. Auch hier ist z.B. eine Zusammenfassung vorgesehen.
- `letter` ist für Anschreiben gedacht.

Oben habe ich mich für die documentclass `book` entschieden und die folgenden Optionen festgelegt:

- `11pt` legt als Schriftgrad 11 Points fest. Der Standardwert liegt bei 10 pt. Wenn der Ihnen genügt, brauchen Sie also gar keine Angabe zu machen.
- `a4paper` legt fest, dass das Seitenformat Din A4 ist. Außer bei der Ausgabe als pdf-Dokument ist ein amerikanisches Seitenformat hier der Standard.
- `oneside` legt fest, dass alle Seiten an der gleichen Stelle nummeriert werden. Alternativ dazu können Sie `twoside` wählen. Dann werden Seiten abwechselnd links und rechts mit einer Nummer versehen.
- `draft` ist eine praktische Option, da sie dafür sorgt, dass Zeilen, an denen Ein Wort in den Seitenrahmen hineinragt mit einem schwarzen Quadrat gekennzeichnet werden. Das ist in sofern praktisch, als Sie schneller sehen, wo Sie ein wenig Feintuning beim Zeilenumbruch machen müssen.

usepackage

Dieser LaTeX-Container ermöglicht es Ihnen Packages zu nutzen, mit denen Sie bestimmte Formatierungen anpassen können. Zum Teil erhalten Sie

dadurch neue Container, die im „Standard-“ LaTeX nicht enthalten sind.

Wenn Sie mehrere Packages nutzen wollen, ohne nähere Optionen auszuwählen, dann können Sie diese durch Kommata getrennt gemeinsam als Argument eines usepackage-Containers verwenden. Bsp.: `\usepackage{palatino, url}` fügt die Packages palatino und url hinzu. palatino ist ein Schriftsatz und url ist ein Container, mit dem Sie URLs im fließenden Text hervorheben können.

Wenn Sie dagegen bei einem Package Optionen festlegen wollen, dann müssen Sie für jedes Package einen eigenen Container programmieren. Bsp.: `\usepackage[ngermanb]{babel}` fügt das babel-Package hinzu, das die Syntaxprüfung für verschiedene Sprachen ermöglicht. Hier müssen wir eine Option wählen, da wir uns für eine Sprache entscheiden müssen. Die Option german entspricht allerdings der alten Rechtschreibung, weshalb mit `ngerman` eine eigene Option für die seit Ende der 90er Jahre geltenden Rechtschreibregeln entwickelt wurde.

Das Package `inputenc` kann die verwendete Codierung festlegen. Darüber hatten wir ja bereits bei HTML gesprochen, der entsprechende usepackage-Container sollte damit klar sein.

setlength

Der Container `\setlength{\parindent}{0cm}` ist dann ein Beispiel dafür, dass wir die Vorstellung eines Containers aus HTML nicht direkt in LaTeX übertragen können: Hier haben wir ein „Tag“, das zwei Inhalte hat, zum einen `\parindent`, was für den Einzug der ersten Zeile eines Absatzes steht, zum anderen `0cm` was naheliegender Weise für 0 Zentimeter steht.

Absätze beginnen bei LaTeX mit einem kleinen Einzug in der ersten Zeile und können nur anhand dieses Einzugs erkannt werden. Wenn Sie dagegen Absätze dadurch trennen wollen, dass Sie eine leere Zeile einfügen wollen und keinen Einzug haben wollen, dann müssen Sie in der Präambel diese Zeile einfügen.

7.2 Text, Sonderzeichen und Formeln

Fast alles, was wir von jetzt an kennen lernen wird im body des Dokuments programmiert. Dieser wird ähnlich einem HTML-Container als `\begin{document}` begonnen und mit `\end{document}` beendet.

Wenn Sie dort einen einfachen Text verfassen wollen, dann können Sie direkt damit beginnen: Anders als in HTML gibt es keine expliziten Absatz-Container wie `<p></p>` in LaTeX.

Das Ende eines Absatzes „programmieren“ Sie dadurch, dass Sie einfach die Enter-Taste drücken. Wollen Sie zusätzliche Leerzeilen einfügen, dann müssen Sie `\\` eingeben.

7.2.1 Sonderzeichen

Im wissenschaftlichen Bereich nutzen wir immer wieder Sonderzeichen, die dann bei einer Textverarbeitung wie LaTeX nicht direkt eingegeben werden kann, weil dieses Zeichen dort eine besondere Bedeutung hat. Den Backslash und einige mehr haben Sie schon kennen gelernt. Wenn Sie ein solches Zeichen verwenden wollen oder Teile von Programmtexten einfügen wollen, gibt es unter anderem zwei einfache Möglichkeiten. Die beiden folgenden Varianten funktionieren für alle Sonderzeichen identisch:

- Wenn Sie innerhalb eines Absatzes einzelne Sonderzeichen oder kurze Textpassagen mit Sonderzeichen einbinden wollen, dann nutzen Sie dazu `\verb|Text mit Sonderzeichen|`. Der `\verb`-Container hat eine Besonderheit: Sie können hier jedes Zeichen (also nicht nur geschweifte Klammern) nutzen, um ihn abzugrenzen.

  Wenn Sie also das Sonderzeichen `|` verwenden wollen, dann nehmen Sie einfach ein anderes, um den Rahmen des `\verb`-Containers festzulegen. Bsp.: `\verb ~ Text mit Sonderzeichen ~` (Hier wurde `~` als Zeichen verwendet, um den Inhalt des Containers abzugrenzen.)

- Wenn Sie dagegen mehrere Zeilen mit Sonderzeichen anzeigen lassen, dann nutzen Sie die sogenante verbatim-Umgebung:

  ```
  \begin{verbatim}
  Zeile mit Sonderzeichen
  Noch eine Zeile mit Sonderzeichen
  Noch viele Zeilen mit Sonderzeichen
  ...
  \end{ verbatim}
  ```

  Anmerkung: Sollten Sie den seltenen Fall haben, dass Sie innerhalb einer verbatim-Umgebung `\end{verbatim}` eintragen wollen, dann lassen Sie einfach eine Leerstelle zwischen der geschweiften Klammer `{` und `verbatim}` stehen.

7.2.2 Formeln

In vielen Texten wird erklärt, dass Sie Formeln in LaTeX mit dem Dollar-Symbol `$` abgrenzen. Das funktioniert zwar, allerdings handelt es sich dabei um eine TeX-Anweisung. In LaTeX gibt es für Formeln im Fließtext folgende Zeichensequenz:

```
\( ... Formel ... \)
```

Dabei gilt, das alles, was zwischen `\(` und `\)` steht im Sinne des mathematischen Modus interpretiert wird. Der mathematische Modus bietet außerordentlich vielfältige Möglichkeiten, um Formeln einzutragen. Wenn Sie in irgend einem mathematischen Buch ein Symbol sehen, dass in einer Formel verwendet wird, dann gibt es ein LaTeX-Tag, mit dem Sie dieses Symbol im mathematischen Modus erzeugen können.

Wie Sie sich vorstellen können, würde eine Einführung in den mathematischen Modus alleine schon ein Buch füllen. An dieser Stelle werde ich nur auf einige wenige Möglichkeiten eingehen, die Ihnen bei den ersten Schritten helfen werden. Alles weitere können Sie in aller Regel durch eine Recherche im Netz sehr schnell in Erfahrung bringen.

- Für Multiplikationen sollten Sie `\cdot` nutzen. Dadurch wird ein Multiplikationspunkt eingefügt.
- Einen Bruch können Sie mit `\frac{Divident}{Divisor}` darstellen. Divident und Divisor können dabei beliebig komplexe Formeln sein.
- Um eine Potenz darzustellen benutzen Sie den Hochpfeil `^`. Wie immer gilt: Wenn der Exponent ein Ausdruck ist, dann nutzen Sie die geschweiften Klammern. Bsp.: x^{2+3} programmieren Sie als `x ^{2 + 3}`.
- Indizes wie $X_{i,j}$ stellen Sie durch `X_{i, j}` dar.

Wenn Sie dagegen mehrere Formeln als eigenständige Absätze anzeigen lassen wollen, dann nutzen Sie dafür die equation-Umgebung:

```
\begin{equation}
Eine Zeile für jede Zeile der Formel.
In dieser Umgebung gilt wieder der mathematische Modus,
wie Sie ihn gerade kennen gelernt haben.
\end{equation}
```

7.3 Umgebungen

Gerade haben Sie mit `\begin{verbatim}` und `\end{verbatim}` etwas kennen gelernt, das konzeptionell den Containern in HTML entspricht. Bei LaTeX werden diese Container aber als **Umgebung**en bezeichnet.

Wenn Sie beim Anfangs-„Tag“ einer Umgebung Optionen programmieren, dann gilt hier dasselbe, was Sie bei den Attributen bzw. Attributen mit Wertzuweisung in HTML kennen gelernt haben: Diese gelten für alles, was sich innerhalb der Umgebung befindet.

7.4 Inhaltsverzeichnis, Kapitel und Abschnitte

Wenn Sie längere Texte als einen Brief schreiben, dann benötigen Sie noch Möglichkeiten, um z.B. Kapitelüberschriften einzufügen. Aus diesen Überschriften wird später übrigens das **Inhaltsverzeichnis** an genau der Stelle erzeugt und ins Dokument eingefügt, an der Sie `\tableofcontents` ins Dokument eintragen.

7.4.1 Kapitel, Abschnitte usw.

Bezüglich der Größe und Strukturierung anhand von Überschriften ist LaTeX komfortabler als HTML: Während dort `<h1>` und `<h2>` nur optisch unterschiedlich sind, bewirken `\chapter{Titel}` und `\section{Titel}`, dass der eine von LaTeX als Teil des anderen interpretiert wird.

Wichtig:

Diese Überschriften sind keine Umgebungen, sondern werden wie abgeschlossene Container in HTML programmiert und behandelt. Das bedeutet, dass der Text und die Unterüberschriften z.B. innerhalb eines Kapitels für LaTeX nicht Teil des Kapitels sind. Auch wenn das in aller Regel kein Problem ist müssen wir deshalb selbst darauf achten, welche Teile unserer Texte zu welchem Kapitel gehören.

Hier nun die Hauptstrukturüberschriften in LaTeX:

- `\part[Kurztitel]{Titel}` wird in aller Regel nur bei Büchern genutzt. Es handelt sich hier um eine Überschrift, die den Inhalt mehrerer Kapitel zusammenfasst. (Denken Sie an so etwas wie die Unterteilung eines Mathebuchs in `Teil 1 - Algebra`, `Teil 2 - Geometrie` usw.)
- `\chapter` entspricht einem Kapitel.

- `\section` entspricht einem Abschnitt innerhalb eines Kapitels.
  - Um Unterabschnitte zu beginnen, nutzen Sie `\subsection`.
  - Dann gibt es noch die `\subsubsection` usw.
- `\paragraph` ist ein Absatz innerhalb eines Abschnitts, der eine eigene Überschrift erhalten soll. Auch hier können sie mit dem Präfix `sub` wie bei sections eine Priorisierung erstellen. Allerdings werden paragraphs nicht ins Inhaltsverzeichnis mit aufgenommen und auch nicht nummeriert.

Übrigens können Sie bei all diesen Typen jeweils in den geschweiften Klammern die Überschrift festlegen, die im Text angezeigt wird.

Und in eckigen Klammern können Sie eine Kurzfassung der Überschrift einfügen, die z.B. im Inhaltsverzeichnis angezeigt wird. Wenn Sie dort nichts angeben, wird auch im Inhaltsverzeichnis die vollständige Überschrift übernommen.

7.5 Auslagern von Kapiteln

Je länger Ihr Dokument wird, desto stärker wird der Wunsch werden, einzelne Teile auszulagern, damit Sie etwas übersichtlicher arbeiten können. In HTML war das nur über den Umweg von PHP möglich, in LaTeX ist es einfacher:

Hier verwenden Sie `\include{Dateiname}`, wobei der Dateiname auf `.tex` enden muss. Diese Endung wird aber in der include-Anweisung nicht aufgeführt, sondern nur der Dateiname vor dem .tex .

7.6 Titelblatt und Glossar

Bei einigen Dokumentklassen ist ein Titelblatt vorgesehen, bei anderen müssen sie über den Eintrag der Option `titlepage` zur `\documentclass` angeben, dass ein Titelblatt hinzugefügt werden soll.

Dass alleine genügt aber noch nicht. Was wir noch brauchen sind die Angaben für das Titelblatt und die Angabe, wo genau das Titelblatt eigefügt werden soll:

Für die nötigen Angaben fügen Sie am Anfang des body (also direkt nach `\begin{document}`) die folgenden LaTeX-Tags ein:

- `\title{}` enthält den Titel, der auf dem Dokument eingeblendet wird.
- `\author{}` enthält den Namen des/der Authoren.
- `\date{}` enthält ein Datum.
  - Dabei können Sie mit `\date{\today}` automatisch das aktuelle Datum eintragen.

Wenn Sie diese Angaben eingetragen haben, können Sie per `\maketitle` an beliebigen Stellen im Dokument ein Titelblatt erzeugen und einfügen lassen.

7.7 Glossar

Um ein Glossar bzw. Stichwortverzeichnis zu generieren müssen Sie leider deutlich mehr Arbeit aufwenden, aber wenn Sie das getan haben, wird genau wie beim Inhaltsverzeichnis ein Verzeichnis für Schlagwörter automatisch generiert und aktualisert.

1. `\makeindex` muss zur Präambel hinzugefügt werden.
2. `\usepackage{makeidx}` kann zusätzlich in der Präambel aufgenommen werden, um die Darstellung des Stichwortverzeichnisses anders zu gestalten.

Wenn Sie das erledigt haben, müssen Sie an der Stelle des Dokuments, wo das Stichwortverzeichnis eingefügt werden soll, die folgenden Zeilen einfügen:

- `\renewcommand{\indexname}{Stichwortverzeichnis}` legt den Titel des Stichwortverzeichnisses fest. (Es gibt eine Standardbezeichnung.)
- `\addcontentsline{toc}{chapter}{Stichwortverzeichnis}` stellt sicher, dass das Stichwortverzeichnis ins Inhaltsverzeichnis aufgenommen wird.
- `\printindex` fügt an der Stelle, an der es steht das Stichwortverzeichnis in das Dokument ein.

7.7.1 Stichwörter und Bezüge festlegen

Jetzt kommt der Teil, der die eigentliche Arbeit für das Glossar ausmacht: Jeder Begriff, der im Glossar aufgenommen werden soll muss mit `\index{Bezeichnung im Glossar}` aufgenommen werden.

Bsp.: Sie wollen einen Verweis auf eine Textpassage festlegen, in der es um die Nutzung von Dampfmaschinen im Allgemeinen geht. Das sähe dann so aus:

```
Bis zu einem dem Autor nicht bekannten und
aus Faulheit nicht recherchierten Zeitpunkt
waren Dampfmaschinen\index{Dampfmaschine}
eine recht weitverbreitete Antriebsart. ...
```

Wenn Sie dabei Haupt- und Unterstichworter nutzen wollen, dann sieht das so aus: `\index{Hauptstichwort!Unterstichwort}`.

Bsp.: Sie schreiben über die Programmierung in C und wollen einen entsprechenden Verweis im Stichwortverzeichnis erzeugen. Das sähe so aus:

```
C\index{Programmiersprache!C} ist eine
kompilierte Sprache ...
```

Doch während das Inhaltsverzeichnis von LaTeX automatisch generiert wird, müssen wir die Generierung des Glossars manuell starten. TeXStudio hat dafür einen Assistenten, den sie per `F12` oder über den entsprechenden Eintrag im Menü `Assistenten` starten können.

7.7.2 Weitere Verzeichnisse

Später werden Sie erfahren, wie Sie Abbildungen und die sogenannten Figures in LaTeX programmieren können. Für diese können Sie ähnlich wie beim Inhaltsverzeichnis Verzeichnisse automatisch generieren lassen. Dazu nutzen Sie `\listoffigures` und `listoftables`.

7.8 Listen und Tabellen

Immer wenn Sie mehrere Einträge gruppieren und/oder sortieren wollen, nutzen Sie sogenannten Listen bzw. Tabellen.

Aus HTML kennen Sie die unordered und ordered lists sowie die description lists. Und alle drei werden (wenn auch mit einer etwas anderen Syntax) nahezu identisch in LaTeX umgesetzt:

Einzelne Einträge werden durch ein vorangestelltes `\item` gekennzeichnet. Als Option können Sie hier noch Labels vergeben.

Alle Listen werden als Umgebung (also mit `\begin{}` und `\end{}`) programmiert. Die Argumente lauten dabei:

- itemize (für Listen mit Punkten)
- enumerate (für nummerierte Listen)
- description (für Glossare)

7.8.1 Tabellen

Hier haben wir einen Unterschied gegenüber HTML: Dort ist eine Tabelle ein Rahmen, anhand dessen wir Elemente unsers Dokuments an einer festen Position im Dokument anordnen können.

In LaTeX dagegen wird zwischen einem table und einem tabular matter unterschieden. Das was wir umgangssprachlich als Tabelle bezeichnen ist in LaTeX die **tabular matter**.

Die unterschiedlichen Bezeichnungen basieren darauf, dass LaTeX hier den Begriff Tabelle so nutzt wie das bei Schriftsetzern üblich ist. Die differenzieren hier genauer als wir das umgangssprachlich tun.

Um klar zwischen umgangssprachlicher Tabelle und den genannten formalen Tabellen bzw. tabellarischen Aufstellungen zu unterscheiden wird hier bei letzteren die englische Bezeichnung gewählt.

Hinweis, wenn Sie mehr dazu im Netz recherchieren wollen: Leider ist der Begriff des formular table im Englischen doppelt belegt: Dort wird er häufig für eine formale Eindeckung eines Tisches verwendet. Wenn Sie also nach `formal table` suchen, werden Sie fast ausschließlich Anleitungen für Diener finden. Bei den verbleibenden Einträgen steht in aller Regel nur, dass es sich dabei um etwas anderes handelt, als das, was wir umgangssprachlich als Tabelle bezeichnen. Eine Aussage, die uns im Regelfall gar nicht weiterhilft.

formal table

Eine Tabelle (**formal table**) in LaTeX ist dagegen eine Umgebung, die den Titel einer Tabelle und ein Label enthalten muss, auf das wir von anderen Stellen des Dokuments aus verweisen können.

Es handelt sich also um etwas, das weitgehend dem `<figcaption>` -Container in HTML entspricht. Es unterscheidet sich allerdings davon, weil ein formal table (`table`-Umgebung) in LaTeX eine Umgebung für beliebige Inhalte außer für Bilder, Videos und Audiodateien wie in HTML ist.

Hier ein Beispiel:

```
\begin[wie immer optional: Buchstabe, der die
Position des formal table festlegt]{table}
\caption[Kurztitel]{Titel des formal table}
\label{Referenz, entspricht einem Anker in HTML}
...
(Hier kann z.B. ein tabular matter aber auch
beliebiger anderer Inhalt eingefügt werden.)
...
\end{table}
```

Die Position (Optional der table-Umgebung) kann durch einen von vier Buchstaben festgelegt werden:

1. `t` (top), also am oberen Rand der aktuellen Seite
2. `b` (bottom), also am unteren Rand der aktuellen Seite
3. `h` (here), also an genau der Stelle, wo die Umgebung im Dokument eingefügt wurde.
4. `p` (page): Bei dieser Option wird die Tabelle auf einer eigenen Seite angezeigt.

tabular matter

Dieser Bereich wird ähnlich wie der mathematische Modus nur kurz angeschnitten. Wenn Sie mehr über Tabellen in LaTeX wissen wollen, dann recherchieren Sie dazu bitte im Netz.

Wichtig:

Eine tabular-Umgebung ist für Texte gedacht. Wenn Sie Formeln in einer Tabelle gruppieren wollen, nutzen Sie bitte die array-Umgebung (siehe nächster Abschnitt).

```
\begin{tabular}{ausrichtung1, ausrichtung2, ...}
erste Zeile, erste Spalte & erste Zeile,
zweite Spalte & ... \\
zweite Zeile, erste Spalte & zweite Zeile,
```

```
zweite Spalte & ... \\
...
\end{tabular}
```

Eine solche Tabelle wird also durch eine `tabular`-Umgebung definiert. Nach dem ersten Paar geschweifter Klammern folgt ein zweites Paar, in dem für jede Spalte die Ausrichtung definiert wird:

- `l` linksbündig
- `c` zentriert
- `r` rechtsbündig
- `p{breite}` definiert eine maximale Breite einer Spalte

Wichtig:

Es gibt im Gegensatz zu HTML keine automatische Anpassung der Breite von Spalten.

array

Neben dem tabular matter, der für Texte gedacht ist, gibt es noch die array-Umgebung, die für mathematische Formeln gedacht ist. Diese müssen Sie allerdings zusätzlich per `\usepackage{array}` in der Präambel importieren.

Um Missverständnisse zu vermeiden: In einer array-Umgebung brauchen Sie nicht mehr den mathematischen Modus zu aktivieren, denn er ist dort automatisch aktiviert.

7.9 figures

Jetzt kommen wir zu dem, was der `<figcaption>` in HTML entspricht: Eine Umgebung für Bilder in LaTeX. Sie nutzen die `figures`-Umgebung also genauso, wie Sie die `table`-Umgebung für Tabellen und Texte nutzen konnten.

Es stellen sich also zwei Fragen:

- Warum gibt es die `figures`- und die `table`-Umgebungen?
- Wie können wir Bilddateien in LaTeX-Dokumenten einbinden?

Die Antwort auf die erste Antwort ist simpel: Es gibt getrennte Verzeichnisse für formal tables und figures. Und diese Verzeichnisse werden aus den jeweiligen Umgebungen automatisch generiert, wenn Sie `\listoffigures` bzw. `\listoftables` verwenden, um an einer Stelle im Dokument das entsprechende Verzeichnis erzeugen zu lassen.

7.9.1 Bilddateien in LaTeX

Bevor wir uns mit der Einbindung von Bilddateien beschäftigen können, müssen wir uns mit dem Thema `pdfLaTeX` und `LaTeX` beschäftigen. Ersteres ist eine Erweiterung, mit der wir pdf-Dokumente aus LaTeX-Dokumenten erzeugen können. Dennoch gibt es bei beiden einen entscheidenden Unterschied:

- Wenn wir `pdfLaTeX` verwenden, können wir PNG-, JPG- und PDF-Dateien als Bilddateien einbinden.
- Wenn wir dagegen „nur“ `LaTeX` verwenden, können wir ausschließlich EPS-Dateien verwenden.

Um überhaupt Bilddateien einbinden zu können, müssen wir die Präambel erweitern: `\usepackage{graphix}`

Um eine Bilddatei in unserem Dokument anzeigen zu lassen verwenden wir `\includegraphics{Dateiname ohne Endung}`. Das bedeutet, dass der Compiler automatisch nach einer Datei sucht. Wenn wir also sicherstellen wollen, dass wir ein Dokument sowohl mit `pdfLaTeX` als auch mit `LaTeX` konvertieren können, dann müssen wir die Bilddatei mit gleichem Namen einmal als PNG-, JPG- und PDF-Datei und einmal als EPS-Datei im gleichen Verzeichnis speichern wie das .tex-Dokument.

Sie können noch die Breite und Höhe als optionale Argumente `width = ...cm` bzw. `height = ...cm` festlegen. Dabei können Sie auch andere Maße wie z.B. pt verwenden, so lange diese in LaTeX definiert sind.

Wichtig:

Auch bei gleicher Bezeichnung sind diese Maße nicht mit denen in Adobe-Produkten identisch; leider beharrt besagter Konzern darauf eigene Definitionen einzelner Maßstäbe zu verwenden.

Außerdem können Sie noch über Werte wie `.75/coumnwidth` die Größe proportional anpassen. Allerdings bedeutet das nicht, dass (z.B. bei einer JPG-Datei) das Bild gestochen scharf ist. Das ist nur bei einer Vektorgrafik sichergestellt.

7.10 Referenzen und Labels

An ein oder zwei Stellen haben Sie bereits `\label{Text}` gesehen. Das entspricht einem Anker in HTML. Also brauchen wir jetzt noch den „Link“ auf einen solchen Anker. In LaTeX heißen die aber nicht Link, sondern **Referenz**. Eine Referenz programmieren Sie mit `ref{Text}`.

U.a. wegen Labels empfehle ich bei der Erstellung von LaTeX-Dokumenten von Anfang an die Arbeit mit einem erweiterten Editor wie TeXStudio: Dieser zeigt Ihnen alle im Dokument verwendeten Labels an. Und das ist deshalb wichtig, weil es zu Inkonsistenzen kommen wird, wenn Sie zwei Labels gleich bezeichnen.

7.11 Boxen

Wenn Sie sich eine LaTeX-Referenz ansehen, werden Sie immer wieder über Container bzw. Elemente mit `box` im Namen stolpern. Eine Box bezeichnet so etwas wie einen Bereich, der abgeschlossen ist und Inhalte einer (z.B. gedruckten) Seite enthalten kann. Die größte Box entspricht dabei dem Bereich, der auf einer Seite insgesamt bedruckt werden kann. Generell wird aber das, was wir als einzelne Einheit betrachten als Box bezeichnet.

Es gibt beispielsweise die Möglichkeit durch `\fbox{text}` eine Textpassage im laufenden Text mit einem Rahmen zu umgeben.

Weitere Boxen, mit denen Rahmen erzeugt werden können sind `\shadowbox`, `\doublebox`, `\ovalbox` und `Ovalbox`.

Dann gibt es die `\parbox[pos]{width}{text}`, mit der ein Text am oberen `t` oder unteren `b` Rand einer Seite angezeigt werden kann, die die Breite `width` hat und `text` beinhaltet.

Aber auch das waren wieder nur einige ausgewählte Möglichkeiten, um Teile Ihres Dokuments hervorzuheben.

7.12 Abschluss

Damit haben Sie jetzt neben HTML eine weitere Markup Language kennen gelernt. Doch während Sie HTML kaum im Studium nutzen werden, sollten Sie LaTeX so schnell wie möglich verinnerlichen. Es ist für wissenschaftlichen Arbeiten ein international anerkannter Standard.

Kapitel 8

Gestaltung mit CSS

Kapitel 9

Funktionalität mit PHP 5.6

Kapitel 10

Langfristige Datenspeicherung mit MySQL 5.6

Stichwortverzeichnis