Hochschule
Bremerhaven

Fachbereich II
Management und Informationssysteme
Wirtschaftsinformatik B.Sc.
Informatik B.Sc.

How-to LaTeX

Eine Kurzanleitung

---

von
Fabian Lignitz

Inhaltsverzeichnis

1 Einleitung

In diesem Dokument möchte ich euch die Grundlagen von LaTeX erklären und wie dieses Template genutzt werden kann. Dieses Dokument ist gleichzeitig mit meinem Template erstellt und bietet einen kleinen Überblick, wie Hausarbeiten formal ausgestaltet werden können. Aus diesem Grund hat diese Anleitung ein formales Deckblatt und eine Eigenständigkeitserklärung. Um das Template zu verstehen, schauen wir uns zu Beginn die Verzeichnisstruktur des LaTeX Projektes an, um daraufhin den Aufbau der `hauptdatei.tex` zu verstehen. Anschließend werden wir uns mit den verschiedenen Umgebungen wie `figure`, `table` und `listings` beschäftigen. Dazu werden wir uns auch mit Floatumgebungen in LaTeX auseinandersetzen müssen. Abschließend blicken wir auf die Literaturverwaltung mit `BibTeX`.
Immer wieder werde ich dazu LaTeX Code zeigen und das Ergebnis dieses Codes. Am Ende soll es euch möglich sein, mittels dieses Templates schöne und formal korrekte Abgaben zu schreiben. Neben diesem Template existieren zusätzliche Templates für die Erstellung von Postern, auf welche im Anhang eingegangen wird.
Grundsätzlich unterscheidet sich LaTeX von Word oder Pages dadurch, dass wir Layout und Inhalt nicht zusammenführen. In LaTeX definieren wir unser Layout über Befehle und Umgebungen und schreiben den Inhalt als `plain-text`. So sehen wir beispielsweise, dass sich einige Wörter im obigen Text von anderen unterscheiden. Dies geschieht in diesem Fall durch den Befehl `\texttt{beispieltext}`. Um den klassischen LaTeX Schriftzug zu erhalten, kann man `\LaTeX` nutzen. Es existieren eine Vielzahl solcher Kommandos, wir werden uns nur mit den wenigsten und wichtigsten beschäftigen. Ziel dieser Ausarbeitung soll es nicht sein, dass ihr zu absoluten LaTeX Expert:innen werdet, sondern die Grundlagen beherrscht.
Meine Vorlagen sind so geschrieben, dass sie ohne Probleme in der Informatikinfrastruktur gebaut werden können. Wollt ihr lokal mit diesen Vorlagen arbeiten, müsst ihr LaTeX installieren und braucht zusätzlich noch einige Programme.

2 Die Verzeichnisstruktur

Wenn ihr das Template herunterladet und entpackt (dazu später mehr), dann liegt ein fertiges und kompilierbares LaTeX Projekt vor. Genau genommen liegt dieses Dokument inkl. aller Inhalte vor, lediglich das Deckblatt unterscheidet sich etwas von der Version, die ich euch zum Download anbiete. Ihr könnt also im Detail nochmal nachschauen, wie bestimmte Beispiele in LaTeX aussehen. Wenn das `tar` entpackt wird, liegt folgende Verzeichnisstruktur vor:

```
/
├── beispiele.tex
├── bibtex/
│   └── hauptdatei.bib
├── buildlualatex.sh
├── hauptdatei.pdf
├── hauptdatei.tex
├── log/
└── src
    ├── abbildungen
    │   └── logoneu.png
    ├── anhang/
    │   ├── abbildungen/
    │   ├── anhang.tex
    │   └── anhang_deckblatt.tex
    ├── basic_structure/
    │   ├── abkuerzungen.tex
    │   ├── deckblatt.tex
    │   ├── erklaerung.tex
    │   └── trennung.tex
    ├── einleitung.tex
    ├── fazit.tex
    ├── hauptabschnitt_3.tex
    └── struktur.tex
```

Abbildung 2.1: Verzeichnisstruktur

In der `beispiele.tex` finden sich kleine snippets zu Tabellen, Listings, Abbildungen usw. Seht es als kleinen Spickzettel für den Alltag, für unser Projekt hat diese Datei erstmal keine Bedeutung.
Das `bibtex` Verzeichnis und die darin enthaltende Datei `hauptdatei.bib` sind Teil der Literaturverwaltung von LaTeX, die wir uns später noch anschauen werden.
Das `buildlualatex.sh` Skript enthält den Programmaufruf zum Kompilieren unseres Projektes. Dieses Skript könnt ihr unverändert lassen, es enthält bereits alle notwendigen Befehle und kann zum Bauen einfach mittels `./buildlualatex.sh` ausgeführt werden.
Die `hauptdatei.pdf` ist das Resultat unseres Bauprozesses, dieses PDF lest ihr gerade. In der `hauptdatei.tex` werden alle Pakete eingebunden, das Grundlayout definiert, die Verzeichnisse gebaut und vor allem der eigentliche Inhalt eingebunden. Diese Datei werden wir uns im nächsten Abschnitt ausführlich anschauen.
Das `log` Verzeichnis enthält alle Log- und Metadateien, die beim Bauprozess erstellt werden, auch hiermit müssen wir uns nicht allzu viel beschäftigen.

Das src Verzeichnis enthält den eigentlichen Inhalt unseres PDFs. Es ist strukturiert durch die Verzeichnisse abbildungen, für alle Grafiken, welche wir nutzen wollen, anhang für alle Inhalte und Abbildungen des Anhangs, basic_structure für das Einbinden von Standarddateien wie Deckblatt, Selbstständigkeitserklärung, Trennvorschläge usw. und die .tex Dateien in welche unser eigentlicher Inhalt kommt. Dieser Text steht bspw. in der Datei struktur.tex.

Wenn ihr in diesem Projekt arbeitet, werdet ihr zum größten Teil eure eigenen .tex Dateien und Abbildungen in src hinterlegen. Damit diese auch eingebunden und die korrekten Daten auf dem Deckblatt hinterlegt werden, widmen wir uns aber zunächst der hauptdatei.tex.

3 Die Hauptdatei

Die hauptdatei.tex gliedert sich grob in vier Abschnitte: die Definition der Pakete, des Deckblattes und Zitierstils sowie die eigentliche Einbindung des Inhaltes. Alle Bereiche wollen wir uns im Folgenden einmal anschauen.

3.1 Die Präambel

Fast alle Funktionen und Features sind in LaTeX in sogenannten packages enthalten, welche ähnlich wie in Java importiert bzw. eingebunden werden müssen. In der Präambel werden all diese Pakete definiert. Zusätzlich wird in der Präambel die allgemeine Dokumentenklasse festgelegt. Also ob wir gerade einen Brief, ein Buch oder einen Artikel schreiben. Die Präambel beginnt dabei mit der \documentclass und endet mit Beginn des eigentlichen Dokumentes, also \begin{document}. Die sog. documentclass sieht für dieses Dokument folgendermaßen aus:

```
1 \documentclass[a4paper,12pt,parskip=half,headsepline,DIV=12,numbers=noenddot]{scrar
  ↪ tcl}
```

Listing 3.1: Die documentclass

Wie wir sehen, wird in der documentclass z. B. das Format (A4) und die Schriftgröße (12pt.) definiert. Ebenso legen wir mit scrartcl fest, dass wir einen Artikel schreiben. Es handelt sich dabei um eine sog. KOMA-Script Klasse. Diese Klassen sind in der LaTeX Welt im Prinzip Standard und ein Blick in die Dokumentation lohnt sich allemal. Ebenso werden nicht, wie aus Word bekannt, Seitenränder festgelegt, sondern Anhand des sog. Satzspiegels eine optimale Verteilung von Text zu Seitengröße ermittelt. Hierfür dient die Angabe DIV=12, auch hier empfehlt sich die Dokumentation der KOMA-Script Klassen.

Zu Beginn sollte die `documentclass` unverändert bleiben, kann jedoch mit einiger Erfahrung an die eigenen Bedürfnisse angepasst werden. Auf die `documentclass` folgt das Einbinden von Packages durch den Befehl `\usepackage`. Für die Darstellung der Verzeichnisstruktur in Abbildung 2.1 wird bspw. das Package `dirtree` genutzt. Einige Packages sind in der `hauptdatei.tex` mit kurzen Kommentaren versehen. Auch zu den Packages finden sich Dokumentation, welche immer wieder zurate gezogen werden können und sollten. Neben der Einbindung der Packages können diese auch direkt definiert werden. So wird z. B. das Package `minted` für die Darstellung von Quellcode (siehe z. B. Listing 3.1) genutzt und das Layout direkt in der Präambel festgelegt:

```
\usepackage[outputdir=log, newfloat]{minted}
\setminted{
  frame=lines,
  framesep=2mm,
  baselinestretch=1.2,
  bgcolor=LightGray,
  fontsize=\footnotesize,
  linenos,
  breaklines=true,
  breakanywhere=true,
  autogobble,
  tabsize=2
}
```

Listing 3.2: Package Definition

Ich werde an dieser Stelle nicht auf einzelne Packages und Einstellungen eingehen, empfehle für einen genauen Einblick aber ausdrücklich die einzelnen Dokumentationen. Auch hier gilt, dass die üblicherweise gebrauchten Packages eingebunden und definiert sind und nur in Sonderfällen zusätzliche Packages gebraucht werden.

3.1.1 Der Zitierstil

Auch der Zitierstil wird innerhalb der Präambel definiert. Wir werden uns in einem späteren Kapitel ausführlicher mit Literaturverwaltung und Zitaten in LaTeX beschäftigen. Aktuell soll uns das Festlegen eines Zitierstiles ausreichen.
In der `hauptdatei.tex` sind bereits zwei Zitierstile definiert, es muss nur einer ausgewählt werden. Die Stile `ieee` und `apa` sind dabei in der Informatik gängig, entscheidet entsprechend eurem eigenen Geschmack und natürlich nach Vorgabe eurer Lehrenden.

```
% Zitierung nach IEEE

\usepackage[
backend=biber,
style=ieee,
autocite=inline,
]{biblatex}
\addbibresource{bibtex/hauptdatei.bib}

% Zitierung nach APA

%\usepackage[
%backend=biber,
%style=apa,
%autocite=inline,
%]{biblatex}
%\addbibresource{bibtex/hauptdatei.bib}
```

Listing 3.3: Auswahl der Zitierstile

Um den Stil zu verändern, könnt ihr den jeweils anderen Stil mit einem % ein- bzw. auskommentieren. Aktuell ist der Stil `ieee` ausgewählt. Welche Unterschiede zwischen den Stilen bestehen, kann in den jeweiligen Guidelines nachgeschaut werden.

3.1.2 Das Deckblatt

Das Deckblatt ist unter `./src/basic_structure/deckblatt.tex` definiert. In vielen Fällen können die nötigen Parameter jedoch direkt in der Präambel festgelegt werden.

```
% true für Bachelorarbeit / false für Hausarbeit
\newbool{bachelorarbeit}
\setbool{bachelorarbeit}{true}

% Setze Fachbereich
\newcommand{\department}{Fachbereich II \\ Management und Informationssysteme}

% Setze Studiengang
\newcommand{\studyprogram}{Wirtschaftsinformatik B.Sc.}

```

```
11 % Setze Modulname (bachelorarbeit muss false sein)
12 \newcommand{\modulname}{Modulname}
13
14 % Setze Dozent:in (bachelorarbeit muss false sein)
15 \newcommand{\auditor}{\textbf{Dozent:in:} \> Prof. Dr. Maxi Muster}
16
17 % Setze Gutachter:innen (bachelorarbeit muss true sein)
18 \newcommand{\firstauditor}{\textbf{Erstgutachter:} \> Prof. Dr. Maxi Mustermann}
19 \newcommand{\secondauditor}{\textbf{Zweitgutachterin:} \> Prof. Dr. Maxi Musterfrau}
20
21 % Setze Titel und Untertitel der Abreit
22 \newcommand{\thetitle}{Eine unnötig komplizierte und sehr sehr sehr lange
   ↪  Abhandlung}
23 \newcommand{\thesubtitle}{Untersuchung einer Sache von wirklichem Wert}
24
25 % Setze Autor:in und MatNr.
26 \newcommand{\theauthor}{Maxi Mustermensch}
27 \newcommand{\matriculationnumber}{00000}
28
29 % Abstand zwischen Name und MatNr. (siehe Deckblatt)
30 \newcommand{\myspace}{1.0cm}
31
32 % Muss in src/basic_structure/deckblatt.tex einkommentiert werden!
33
34 \newcommand{\secondauthor}{\> Maxi Mustermensch \> MatNr. 00000\\}
35 \newcommand{\thirdauthor}{\> Maxi Mustermensch \> MatNr. 00000\\}
36 \newcommand{\fourthauthor}{\> Maxi Mustermensch \> MatNr. 00000\\}
37 \newcommand{\fifthauthor}{\> Maxi Mustermensch \> MatNr. 00000\\}
```

Listing 3.4: Definition des Deckblatts

Viele der Angaben sollten selbsterklärend sein und können sonst durch simples ausprobieren weiter erforscht werden. Beachtet jedoch, dass die Sonderzeichen \\ und \> erhalten bleiben müssen, diese sind wichtig für das Layout. Ebenso sollte Zeile 30 bzw. die Angabe des `\myspace` bei mehreren Autor:innen beachtet werden. Der Wert steht für den Abstand zwischen Name und MatNr. und muss sich damit entsprechend am längsten Namen orientieren. Sollte eure Abgabe mehr als eine Autor:in haben, können die Werte für weitere Autor:innen direkt in der Präambel gesetzt werden, müssen jedoch zusätzlich in der `deckbatt.tex` einkommentiert werden.

Aus den in dem Listing 3.4 gewählten Einstellungen ergibt sich folgendes Deckblatt:

Abbildung 3.1: Deckblatt

Das Deckblatt ist manuell gesetzt und sollte nicht ohne entsprechendes Know-how angepasst werden. Natürlich könnt ihr immer etwas rumprobieren und euch zur Not das Projekt neu herunterladen.

3.2 Den Inhalt einbinden

Wie bereits erwähnt, endet mit `\begin{document}` die Präambel, damit beginnt unser eigentlicher Inhalt. Der Befehl `\begin{document}` öffnet dabei eine Umgebung oder `environment`, welche durch `\end{document}` geschlossen wird. Dies ähnelt etwas dem `if` und `fi` aus der Bash und bildet einen Block.

Zu Beginn werden die Kopf- und Fußzeile des Dokuments beschrieben:

```
1  % Definition Header Sections sollen in der Kopfzeile stehen; Kopfzeile mit
   ↪  Unterstrich
2  \automark[subsection]{section}
3  \KOMAoptions{headsepline=true}
4  %\ihead{Kopfzeile innen}
5  %\chead{Kopfzeile außen}
6  \ohead{\headmark}
7
8  % Definition footer \pagemark steht für Seitennummer
9  %\ifoot{Fußzeile innen}
10 %\cfoot{Fußzeile Mitte}
11 \ofoot{\pagemark}
```

Listing 3.5: Kopf- und Fußzeile

Daraus ergibt sich, wie in diesem Dokument zu sehen, eine Kopfzeile, welche die Überschrift (`section`) oder den Unterabschnitt (`subsection`) enthält und durch eine Linie abgegrenzt wird. Dabei wird immer nur die `section` bzw. `subsection` aufgeführt, welche als Erstes auf der Seite beginnt. Es existieren hier einige mögliche Definitionen, welche in der Dokumentation des Packages `scrlayer-scrpage` nachgelesen werden können. Die Fußzeile wird schlicht mit der aktuellen Seitennummer definiert.
Auf die Definition der Kopf- und Fußzeile folgt das Einbinden von Trennvorschlägen. LaTeX verfügt grundsätzlich über eine sehr gute automatische Silbentrennung, falls diese aber mal versagt, können eigene Trennvorschläge eingebunden werden. Beispiele hierzu sind in der `trennung.tex` hinterlegt.
Auf weitere kleinere Einstellungen und optionale Layoutvarianten, welche ausreichend kommentiert sind, folgt die Einbindung des Deckblattes, mit dem wir uns ja schon beschäftigt haben. Die nun folgenden `inputs` sorgen dafür, dass euer geschriebener Text auch in das Dokument eingebunden und angezeigt wird. So sorgen die folgenden Inputs dafür, dass der bisherige Text, den ihr in diesem Dokument lesen konntet, eingebunden wird.

```
\input{src/einleitung.tex}
\input{src/struktur.tex}
\input{src/zurhauptdatei.tex}
```

Listing 3.6: input statements

Auf die `inputs` folgen die jeweiligen Verzeichnisse, auch hier sollten die Einstellung nicht ohne entsprechende Kenntnisse geändert werden. Grundsätzlich werden die Verzeichnisse automatisch erstellt, lediglich für den Fall, dass ein Verzeichnis leer bleiben würde, da das Dokument z. B. keine Tabellen enthält, muss der entsprechende Teil der `hauptdatei.tex` auskommentiert werden. Dabei kann es passieren, dass sich die Kopfzeile nicht mehr korrekt verhält, da diese teilweise für die Verzeichnisse manuell angepasst werden muss.
Das Abkürzungsverzeichnis stellt eine Besonderheit dar, dieses muss manuell befüllt werden, dazu gibt es Beispiele in der `abkuerzungen.tex`.
Auf die Verzeichnisse folgt das Deckblatt für den Anhang und der Anhang selbst. Dort muss lediglich beachtet werden, dass für Überschriften auf höchster Ebene nur `subsections` verwendet werden dürfen, da der Anhang selbst die `section` darstellt.
Oftmals sieht man in LaTeX, dass die Präambel in eine eigene Datei ausgelagert wird. Dies dient jedoch nur der Übersichtlichkeit und der etwas klareren Trennung, ich verzichte darauf in dieser Vorlage.

4 Den Inhalt erstellen

Wie bereits erwähnt, wird Text in LaTeX als `plain-text` geschrieben. Was bedeutet, dass wir einfach Text schreiben können und dieser wird in der definierten Schriftart und -größe dargestellt. Zur Veranschaulichung soll uns folgendes kleines Textbeispiel dienen:

Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte!

Der oben stehende Text sieht in der `.tex` Datei folgendermaßen aus:

```
Er hörte leise Schritte hinter sich.
Das bedeutete nichts Gutes.
Wer würde ihm schon folgen, spät in der Nacht
und dazu noch in dieser engen Gasse mitten
im übel beleumundeten Hafenviertel?
```

```
6 Gerade jetzt, wo er das Ding seines Lebens gedreht hatte
7 und mit der Beute verschwinden wollte!
```

Listing 4.1: Textbeispiel

Wir sehen, der Inhalt der beiden Ansichten ist identisch, das Layout jedoch nicht. In der Datei ist der Text auf sieben Zeilen verteilt, im fertigen Dokument wird allerdings nicht automatisch ein Umbruch eingefügt. Einen solchen Umbruch müssen wir manuell setzen:

Er hörte leise Schritte hinter sich.
Das bedeutete nichts Gutes.
Wer würde ihm schon folgen, spät in der Nacht
und dazu noch in dieser engen Gasse mitten
im übel beleumundeten Hafenviertel?
Gerade jetzt, wo er das Ding seines Lebens gedreht hatte
und mit der Beute verschwinden wollte!

In der Datei sieht dies folgendermaßen aus:

```
1 Er hörte leise Schritte hinter sich.\\
2 Das bedeutete nichts Gutes.\\
3 Wer würde ihm schon folgen, spät in der Nacht\\
4 und dazu noch in dieser engen Gasse mitten\\
5 im übel beleumundeten Hafenviertel?\\
6 Gerade jetzt, wo er das Ding seines Lebens gedreht hatte\\
7 und mit der Beute verschwinden wollte!\\
```

Listing 4.2: Textbeispiel 2

Einen Zeilenumbruch wird in LaTeX somit folgendermaßen realisiert: \\

Soll statt einem Umbruch eine Leerzeile folgen, muss auch in der Datei eine Leerzeile eingefügt werden. Wie in diesem Dokument schon mehrfach gesehen und kurz in der Einleitung erwähnt, können Eigennamen oder Wörter, die man hervorheben will, mit dem Befehl `\texttt{Beispieltext}` realisiert werden, dies sieht dann so aus: `Beispieltext`. Um Text fettgedruckt darzustellen, können wir das Kommando `\textbf{Beispieltext}` nutzen, dies sieht dann so aus: **Beispieltext**.
Wenn Text erst einmal unformatiert ausgegeben wird, dann müssen wir uns auch Gedanken machen, wie wir Überschriften darstellen können. Überschriften werden grundsätzlich mittels des Befehls `\section{Überschrift}` realisiert.

5 Überschrift

Abbildung 4.1: Beispiel für eine section

Mittels des Befehls `\subsection{Unterabschnitt}` kann ein Unterabschnitt erstellt werden. Für weitere Abstufungen kommt entsprechend der Befehl `\subsubsection{}` zum Einsatz.

5.1 Unterabschnitt

Abbildung 4.2: Beispiel für eine subsection

Die Grundlagen für das Textlayout sind damit gegeben. Es gibt noch einige zusätzliche Möglichkeiten z. B. das Einrücken von Text, wie es bei längeren Zitaten üblich ist:

> Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte!

Eine solche Einrückung ist über die `quote environment` möglich. Gleichzeitig möchte man oft eingerückten Text in einer etwas kleineren Schriftgröße darstellen. Dies bringt uns zu der Definition der Schriftgröße. Als Standard dient die in der `documentclass` festgelegte Schriftgröße, von der über Befehle nach oben oder unten abgewichen werden kann.

Tabelle 4.1: LaTeX Schriftgrößen

Befehl	Ergebnis
\tiny	winzig
\scriptsize	klein
\footnotesize	etwas größer
\small	noch etwas größer
\normalsize	normal
\large	groß
\Large	größer
\LARGE	noch größer
\huge	riesig
\Huge	noch riesiger

Damit sieht unser eingerückter Text als LaTeX Code folgendermaßen aus:

```
\begin{quote}
  \footnotesize{
  Er hörte leise Schritte hinter sich.
  Das bedeutete nichts Gutes.
  Wer würde ihm schon folgen, spät in der Nacht
  und dazu noch in dieser engen Gasse mitten
  im übel beleumundeten Hafenviertel?
  Gerade jetzt, wo er das Ding seines Lebens gedreht hatte
  und mit der Beute verschwinden wollte!}\\
\end{quote}
```

Listing 4.3: Textbeispiel 3

Ebenso existieren einige Sonderzeichen, welche nur durch spezielle Befehle dargestellt werden können.

Tabelle 4.2: LaTeX Sonderzeichen

Befehl	Ergebnis
\&	&
\%	%
\$	$
\#	#
\{	{
\}	}
\textasciitilde	~
\textasciicircum	^
\textbackslash	\
\glqq{}	„
\grqq{}	“

Grund ist, dass alle diese Zeichen in LaTeX eine eigene Bedeutung haben. Entsprechend muss unterschieden werden können, ob wir einen LaTeX Befehl ausführen oder das Sonderzeichen im Text ausgeben wollen. Auf eines der Zeichen, die sog. Tilde (~) soll kurz eingegangen

werden. Diese sieht man meistens vor oder nach Befehlen im Code, so z. B. jedes Mal, wenn der LaTeX (`\LaTeX~`) Schriftzug genutzt wird. Die Tilde steht dabei für ein Leerzeichen, welches wir in solchen Fällen explizit angeben müssen, da LaTeX nicht unterscheiden kann zwischen einem Leerzeichen zum Abgrenzen von Fließtext zum Kommando oder einem zu druckenden Leerzeichen. Wenn ihr also bemerkt, dass euch ein Leerzeichen zwischen einer Kommandoausgabe und Fließtext fehlt, nutzt die ~.
Die letzten beiden Kommandos die wir uns zur Formatierung von Fließtext angucken werden, sind `\vspace*{}` und `\hspace*{}`. Das `\vspace*{1.0cm}` Kommando sorgt für vertikalen Weißraum von einem Zentimeter im Dokument:

Beispieltext

Beispieltext

Der `\hspace{}` steht für einen horizontalen Weißraum und wird von mir deutlich seltener genutzt. Ab und an wird dies aber gebraucht z. B. für den Abstand von Namen und MatrNr. auf dem Deckblatt. Zwischen diesen beiden Wörtern habe ich einen `\hspace{1.0cm}` gesetzt:

Beispieltext Beispieltext

Mittels all dieser Kommandos ist es uns nun möglich, Fließtext ansprechend zu formatieren. Unser Dokument besteht jedoch nicht ausschließlich aus Fließtext, sondern aus einer Mischung aus Text, Listings, Tabellen und Abbildungen. Aus diesem Grund werden wir uns nun mit diesen drei Punkten beschäftigen.

4.1 Float und FloatBarrier

Oft werden Abbildungen und Tabellen in sog. Floatumgebungen genutzt. Dabei übernimmt LaTeX die Aufgabe, die Elemente optimal im Fließtext zu positionieren und so einen angenehmen Lesefluss zu gewährleisten. Was in der Theorie sehr praktisch klingt, klappt m. E. in der Praxis nicht immer gut. Oftmals werden die Elemente um mehrere Seiten verschoben und passen so inhaltlich nicht mehr gut mit dem Geschriebenen zusammen. Ebenso verhindert die Floatumgebung, dass Tabellen über Seitengrenzen hinweg dargestellt werden können, dies ist nur logisch, aber oft nicht gewollt. Dennoch bietet die Verwendung auch Vorteile, so ist der Umgang mit captions (also der Abbildungsbeschreibung) oft einfacher und auch werden die Weißräume vor und nach den Inhalten sauber erstellt, sodass ein manuelles `\vspace` nicht nötig ist. In Abwägung dieser Punkte, habe ich mich dazu entschieden, weitestgehend auf Floats zu verzichten und im Zweifel etwas nachzukorrigieren. Da ihr Online aber oft auf diese Beispiele stoßen werdet und ich für klassische Abbildungen (Bilder) auch eine Floatumgebung verwende, soll kurz gezeigt werden, wie diese aussehen.

Für Floats wird immer eine Umgebung bzw. `environment` geöffnet, für Abbildungen sieht diese folgendermaßen aus:

```
\begin{figure}[!ht]
  \includegraphics[width=0.8\textwidth]{abbildungen/Logo.png}
  \caption[Beschreibung]{Beschreibung}
  \label{Beschreibung}
\end{figure}
```

Listing 4.4: Floatumgebung für figures

Die Floatumgebung wird eingeleitet durch `\begin{figure}`, die Zeichenkombination in den eckigen Klammern steht dann für den sog. `float specifier`. Alle Werte haben dabei eine eigene Bedeutung und können entsprechend kombiniert werden.

Tabelle 4.3: float specifier

Specifier	Berechtigung
h	ungefähr an der gleichen Stelle, an der sie im Ausgangstext vorkommt (jedoch nicht genau an dieser Stelle)
t	Positionierung am oberen Rand der Seite
b	Am unteren Rand der Seite platzieren
p	Auf eine spezielle Seite
!	Überschreibt die internen Parameter, die LaTeX zur Bestimmung „guter“ Gleitpositionen verwendet.

Trotz der Angabe `!ht` ist nicht sichergestellt, dass die Abbildung auch wirklich an der gewünschten Stelle oder innerhalb eines gewünschten Bereiches erscheint. Um trotz Floatumgebung genauer festzulegen, in welchem Bereich die Abbildung erscheinen soll, kann das Kommando `\FloatBarrier` genutzt werden. Durch dieses Kommando wird der Bereich, in dem die Abbildung von LaTeX gesetzt wird, begrenzt. Werden die `\FloatBarrier` direkt ober- und unterhalb der Umgebung gesetzt, sollte die Abbildung an der exakten Stelle aus dem Code erscheinen. Wird zusätzlich noch Fließtext in den Bereich aufgenommen, wird die Abbildung innerhalb dieses Bereiches inkl. des Fließtextes optimal gesetzt.

```
\FloatBarrier
Er hörte leise Schritte hinter sich. Das bedeutete
nichts Gutes. Wer würde ihm schon folgen, spät
in der Nacht und dazu noch in dieser engen Gasse mitten
im übel beleumundeten Hafenviertel? Gerade jetzt, wo
er das Ding seines Lebens gedreht hatte und mit der
Beute verschwinden wollte!

\begin{figure}[!ht]
  \includegraphics[width=0.8\textwidth]{abbildungen/Logo.png}
  \caption[Beschreibung]{Beschreibung}
  \label{Beschreibung}
\end{figure}
\FloatBarrier
```

Listing 4.5: Beispiel FloatBarrier

In diesem Beispiel kann die Abbildung nicht außerhalb der FloatBarrier gesetzt werden, jedoch zwischen den Text. So könnte ein Teil des Textes ober- und unterhalb der Abbildung stehen, die ganze Abbildung jedoch auch über oder unter dem Text dargestellt werden. Im zweiten Beispiel sollte die Abbildung an der exakten Stelle aus dem Code auch im fertigen Dokument erscheinen.

```
Er hörte leise Schritte hinter sich. Das bedeutete
nichts Gutes. Wer würde ihm schon folgen, spät
in der Nacht und dazu noch in dieser engen Gasse mitten
im übel beleumundeten Hafenviertel? Gerade jetzt, wo
er das Ding seines Lebens gedreht hatte und mit der
Beute verschwinden wollte!

\FloatBarrier
\begin{figure}[!ht]
  \includegraphics[width=0.8\textwidth]{abbildungen/Logo.png}
  \caption[Beschreibung]{Beschreibung}
  \label{Beschreibung}
\end{figure}
\FloatBarrier
```

Listing 4.6: Beispiel 2 FloatBarrier

Auch für Tabellen existiert eine Floatumgebung, welche jedoch in diesem Template nicht genutzt wird. Die Floatumgebung für Tabellen wird mit `\begin{table}` eingeleitet und entsprechend beendet.

4.2 Abbildungen

Wenn wir Abbildungen, in diesem Fall speziell Bilder, einbinden, dann binden wir diese mit dem Kommando `\includegraphics` ein. Zusätzlich können bzw. müssen weitere Werte mitgegeben werden. Zum einen, wo die einzubindende Datei liegt und zum anderen welche Größe das Bild erhalten soll. Wie beschrieben, kann die Abbildung dann noch in eine Floatumgebung eingebunden werden. Das einfachste Beispiel könnte folgendermaßen aussehen:

```
1 \includegraphics[width=0.8\textwidth]{abbildungen/Logo.png}
```

Listing 4.7: Beispiel für das Einbinden einer Abbildung

Der Wert in den eckigen Klammern beschreibt die Größe der Abbildung, in diesem Fall, hat die Abbildung den Faktor 0.8 der Textbreite. Setzen wir um das Kommando die Floatumgebung, können wir noch weitere Möglichkeiten nutzen:

```
1 \begin{figure}
2   \centering
3   \includegraphics[width=0.8\textwidth]{abbildungen/Logo.png}
4   \caption[Verzeichniseintrag]{Beschreibung}
5   \label{labelname}
6 \end{figure}
```

Listing 4.8: Beispiel 2 für das Einbinden einer Abbildung

Der Befehl `\centering` steht, wenig überraschend, für das horizontale Zentrieren der Abbildung. Das `\caption` Kommando ermöglicht das Setzen der Abbildungsbeschreibung, die Angabe innerhalb der eckigen Klammern steht dabei für den Eintrag im Abbildungsverzeichnis, der Wert in den geschweiften Klammern für die Abbildungsbeschreibung direkt unterhalb der Abbildung. In den allermeisten Fällen sollten die Angaben identisch sein, in solchen Fällen, kann die Angabe für den Verzeichniseintrag (inkl. der Klammern) einfach weggelassen werden.

4.3 Listings

Listings bzw. die Darstellung von Quellcode stellen in diesem Template eine kleine Besonderheit dar. In Dokumentationen oder Onlineforen findet man für Listings oft die `lstlisting` Umgebung, in diesem Template wird diese nicht genutzt, sondern das `minted` Paket. Dieses ermöglicht m. E. deutlich schönere Listings inkl. deutlich umfangreicherem Highlighting. Da `minted` jedoch nicht außerhalb von Floatumgebungen ohne weiteres mit Captions umgehen kann, habe ich mich entschieden selbst eine Umgebung zu schreiben. Diese Umgebung trägt den Namen `code` und kann so nur in diesem Template genutzt werden.

```
\begin{code}{Beschreibung}{label}
  \begin{minted}{bash}
    #!/bin/bash

    number="$1"
    if test $number -eq 0; then
      echo "true"
    else
      echo "false"
    fi
  \end{minted}
\end{code}
```

Listing 4.9: Beispiel für ein Listing

Wie üblich wird die Umgebung mit dem Befehl `\begin{}` eingeleitet. In diesem Fall müssen wir die `code` Umgebung erstellen und innerhalb dieser dann die eigentliche `minted` Umgebung. Für die `code` Umgebung kann direkt die Listingbeschreibung gesetzt werden. In den geschweiften Klammern der `minted` Umgebung muss dann die Sprache z. B. Bash, Java oder Python gesetzt werden, daraufhin folgt der dazustellende Code. Statt den Quellcode innerhalb der `minted` Umgebung direkt zu schreiben, kann auch eine Datei eingebunden werden.

```
\inputminted{latex}{src/listingcodeexample.txt}
```

Listing 4.10: Beispiel 2 für ein Listing

Beachtet, dass `inputminted`, wie in Listing 4.9 zu sehen, innerhalb der `code` Umgebung stehen muss. Wollen wir nur einen Befehl oder kleinen Codeschnipsel zeigen, kann dieser auch direkt im Fließtext angezeigt werden. Dies könnte z. B. so aussehen:

Überschriften können in HTML mittels `<h1>Überschrift</h1>` dargestellt werden.

Im Code sieht dies dann folgendermaßen aus:

```
Überschriften können in HTML mittels \mintinline{html}|<h1>Überschrift</h1>|
↪ dargestellt werden.
```

Listing 4.11: Inline Listings

4.4 Tabellen

Tabellen sind gegenüber Abbildungen und Listings deutlich individueller und entsprechend existieren unzählige Möglichkeiten, Tabellen zu gestalten. Ich werde versuchen, die Grundlagen so zu erklären, dass ihr in den meisten Fällen ohne zusätzlichen Aufwand Tabellen erstellen könnt.
Für Tabellen stehen verschiedene Packages zur Verfügung. In diesem Template wird das `tabularray` Package genutzt. Dieses ermöglicht eine Vielzahl von Einstellungen, besitzt eine sehr einfache Einbindung der `booktabs` Bibliothek und ist sehr gut dokumentiert.
Die erwähnten `booktabs` beschreiben ein besonderes Tabellendesign, welches an Tabellen aus dem wissenschaftlichen Bereich angelehnt ist. Dabei wird auf vertikale und doppelte Linien verzichtet, ebenso müssen die Linien in Bezug auf Dicke und Länge nicht gesondert definiert werden.
Im Folgenden werden wir uns mit zwei Arten von Tabellen in LaTeXanschauen. Zum einen Tabellen, welche nicht mit der Seite umbrechen können und zum anderen Tabellen, welche über Seitengrenzen hinweg dargestellt werden können. Die beiden Umgebungen, die wir hierfür verwenden, sind `talltblr` und `longtblr`. Die `tallblr` Umgebung wird verwendet, um Tabellen auf einer Seite darzustellen und bietet gleichzeitig einen guten Umgang mit Captions und Labels.

```
\begin{center}
  \begin{talltblr}[caption={Beispiel}, label={beispiel}]{colspec={X[c,m] X[c,m]},
  ↪  width=0.4\textwidth}\toprule
    Überschrift 1 & Überschrift 2  \\ \midrule
    Wert 1.1   & Wert 1.2         \\ \cmidrule{1-2}
    Wert 2.1   & Wert 2.2         \\ \cmidrule{1-2}
    Wert 3.1   & Wert 3.2         \\ \cmidrule{1-2}
    Wert 4.1   & Wert 4.2         \\ \cmidrule{1-2}
    Wert 5.1   & Wert 5.2         \\ \bottomrule
```

```
  \end{talltblr}
\end{center}
```

Listing 4.12: Beispielcode für Tabelle

In diesem Beispiel wird zu Beginn eine `center` Umgebung eingeleitet, dies tue ich in fast allen Fällen, da Tabellen in den meisten Fällen zentriert dargestellt werden sollen, dennoch ist dies natürlich optional. Erst jetzt folgt die eigentliche Tabellenumgebung `talltblr`. Auf diese folgt in den eckigen Klammern die Definition der Caption und des Labels. In den geschweiften Klammern definieren wir nun das Layout der Spalten, an dieser Stelle beginnt es etwas kompliziert zu werden.
Die Angabe `colspec={X[c,m] X[c,m]}` definiert zwei Spalten, wobei Spalten vom Typ X für Spalten mit fester Breite und automatischem Zeilenumbruch stehen. Der Wert `c` steht für eine vertikal und die Angabe `m` für eine horizontal zentrierte Ausrichtung der Spalte. Es existieren entsprechend Angaben für eine rechtsbündige (`r`) und eine linksbündige (`l`) Ausrichtung.

Aus dem Code aus Listing 4.12 ergibt sich folgende Tabelle:

Tabelle 4.4: Tabelle mit zwei Spalten

Überschrift 1	Überschrift 2
Wert 1.1	Wert 1.2
Wert 2.1	Wert 2.2
Wert 3.1	Wert 3.2
Wert 4.1	Wert 4.2
Wert 5.1	Wert 5.2

Die Angaben `\toprule`, `\midrule`, `\cmidrule` und `\bottomrule` stehen für die unterschiedlichen Arten von Linien innerhalb der Tabelle. Die zusätzliche Angabe von Ziffern bei der `\cmidrule` steht für die Anzahl der Spalten, die unterstrichen werden sollen. Der Spalteninhalt wird jeweils durch ein & abgegrenzt und jede Zeile mit einem Zeilenumbruch (\\) beendet.

Es können natürlich auch mehr als nur zwei Spalten definiert werden. Ebenso will man für die Überschriften des Öfteren ein leicht abweichendes Layout festlegen. Dies ist selbstverständlich möglich und kann folgendermaßen umgesetzt werden:

```
\begin{center}
  \begin{talltblr}[caption={Tabelle mit drei Spalten},
  ↪  label={beispielzwei}]{width=0.9\textwidth, colspec={X[l,m] X[c,m]
  ↪  X[l,m]}}\toprule
```

```
    \textbf{Spalte 1} & \textbf{Spalte 2} & \SetCell[c=1]{c} \textbf{Spalte 3}
    ↪                                                    \\ \midrule
    a                 & 1                & Ein ausreichend langer Text, damit es
    ↪  zu einem Zeilenumbruch kommt.                       \\ \cmidrule{1-3}
    b                 & 2                & Ein kürzerer Text.
    ↪                                                    \\ \cmidrule{1-3}
    c                 & 3                & Wieder etwas längerer Text in dieser
    ↪  Spalte, auch hier kann es zu einem Zeilenumbruch kommen. \\ \cmidrule{1-3}
    d                 & 4                & Ein Text.
    ↪                                                    \\ \cmidrule{1-3}
    e                 & 5                & Noch etwas zusätzlicher Text.
    ↪                                                    \\ \bottomrule
  \end{talltblr}
\end{center}
```

Listing 4.13: Beispiecode für Tabelle mit drei Spalten

In diesem Beispiel sind die Überschriften der jeweiligen Spalte zum einen mittels `\textbf{}` fett gedruckt worden, zum anderen ist die Überschrift der dritten Spalte nicht wie der übrige Inhalt linksbündig ausgerichtet, sondern zentriert. Hierfür wird das Kommando `\SetCell[c=1]{c}` genutzt. Die Angabe c=1 steht für die Anzahl der folgenden Spalten (in diesem Fall eine) für die das in den geschweiften Klammern definiert Layout (in diesem Fall zentriert) gelten soll.

Die so beschriebene Tabelle wird dann folgendermaßen dargestellt:

Tabelle 4.5: Tabelle mit drei Spalten

Spalte 1	**Spalte 2**	**Spalte 3**
a	1	Ein ausreichend langer Text, damit es zu einem Zeilenumbruch kommt.
b	2	Ein kürzerer Text.
c	3	Wieder etwas längerer Text in dieser Spalte, auch hier kann es zu einem Zeilenumbruch kommen.
d	4	Ein Text.
e	5	Noch etwas zusätzlicher Text.

Es fällt auf, dass die ersten beiden Spalten eigentlich deutlich weniger Platz bräuchten, als sie einnehmen. Dies liegt daran, dass die zur Verfügung stehende Fläche unter allen Spalten gleich aufgeteilt wird. Dieses Verhalten können wir anpassen, indem wir den Spalten ein Größenverhältnis zueinander zuweisen:

```
\begin{center}
  \begin{talltblr}[caption={Tabelle mit drei Spalten und angepasster
  ↪ Spaltenbreite}, label={beispieldrei}]{width=0.9\textwidth, colspec={X[1,l,m]
  ↪ X[1,c,m] X[5,l,m]}}\toprule
    \textbf{Spalte 1} & \textbf{Spalte 2} & \SetCell[c=1]{c} \textbf{Spalte 3}
    ↪                                                    \\ \midrule
    a                 & 1                 & Ein ausreichend langer Text, damit es
    ↪ zu einem Zeilenumbruch kommt.                        \\ \cmidrule{1-3}
    b                 & 2                 & Ein kürzerer Text.
    ↪                                                    \\ \cmidrule{1-3}
    c                 & 3                 & Wieder etwas längerer Text in dieser
    ↪ Spalte, auch hier kann es zu einem Zeilenumbruch kommen. \\ \cmidrule{1-3}
    d                 & 4                 & Ein Text.
    ↪                                                    \\ \cmidrule{1-3}
    e                 & 5                 & Noch etwas zusätzlicher Text.
    ↪                                                    \\ \bottomrule
  \end{talltblr}
\end{center}
```

Listing 4.14: Beispiecode für Tabelle mit drei Spalten und angepasster Spaltenbreite

Die Zahlen innerhalb der Spalten Definition (`colspec={X[1,l,m] X[1,c,m] X[5,l,m]}`) sorgen nun für ein Größenverhältnis von eins zu fünf. Um die passenden Werte zu finden, braucht es ein bisschen Übung und natürlich auch ein paar Versuche. Unsere angepasste Tabelle hat nun eine angenehme und angemessene Verteilung:

Tabelle 4.6: Tabelle mit drei Spalten und angepasster Spaltenbreite

Spalte 1	**Spalte 2**	**Spalte 3**
a	1	Ein ausreichend langer Text, damit es zu einem Zeilenumbruch kommt.
b	2	Ein kürzerer Text.
c	3	Wieder etwas längerer Text in dieser Spalte, auch hier kann es zu einem Zeilenumbruch kommen.
d	4	Ein Text.
e	5	Noch etwas zusätzlicher Text.

Grundsätzlich können wir aus allen unseren Tabellen einfach seitenübergreifende Tabellen machen, indem wir statt `talltblr` einfach `longtblr` verwenden.
Wir können alle unsere Einstellungen beibehalten und müssen lediglich die Header unserer Tabelle festlegen. Dies ist nötig, damit die Überschriften auf der neuen Seite wiederholt werden können. So sieht die Tabelle 4.4 als `longtblr` fast genauso aus:

```
\begin{center}
  \begin{longtblr}[caption={Tabelle mit zwei Spalten},
  ↪  label={beispieleins}]{colspec={X[c,m]
  ↪  X[c,m]},rowhead=1,width=0.4\textwidth}\toprule
    Überschrift 1 & Überschrift 2 \\ \midrule
    Wert 1.1      & Wert 1.2      \\ \cmidrule{1-2}
    Wert 2.1      & Wert 2.2      \\ \cmidrule{1-2}
    Wert 3.1      & Wert 3.2      \\ \cmidrule{1-2}
    Wert 4.1      & Wert 4.2      \\ \cmidrule{1-2}
    Wert 5.1      & Wert 5.2      \\ \bottomrule
  \end{longtblr}
\end{center}
```

Listing 4.15: Beispiel für eine longtblr

Lediglich die Angabe `rowhead=1` steht als neue Angabe dafür, dass die erste Zeile die Header bzw. Überschriften enthält und somit bei jedem Seitenumbruch wiederholt werden soll.
Mittels dieser Angaben und etwas Übung sollten sich, ohne allzu viel Aufwand, ansehnliche und funktionale Tabellen erstellen lassen. Ich nutze oftmals direkt die `longtblr` Umgebung, dies ermöglicht aus meiner Sicht ein angenehmeres Layout.

4.5 Labels und Verweise

Wie in verschiedenen Listings gesehen und an einigen Textstellen erwähnt, existieren in LaTeX sog. `labels`. Durch das Labeln von Abbildungen, Tabellen, Listings und Überschriften können wir im Fließtext auf diese verweisen. Der Vorteil besteht darin, dass wir uns keine Gedanken um die Nummerierung von Abbildungen o. ä. machen müssen. Wird auf ein Label verwiesen, dann passt sich dieser Verweis automatisch an, sollte sich die Nummerierung geändert haben.
Wir können, in manchen Fällen müssen wir sogar, Labels für alle Überschriften, Tabellen, Listings und Abbildungen erstellen und somit auf diese referenzieren. Für Tabellen, Abbildungen und Listings habe ich die Labels in vielen Beispielen schon gezeigt, schaut euch hierfür nochmal die entsprechenden Listings aus den vorangegangenen Abschnitten an.
Überschriften können einfach mittels des Befehls `\label{}` zusätzlich mit einem Label versehen werden. So ist dieser Abschnitt folgendermaßen mit einem Label ausgestattet:

```
1 \subsection{Labels und Verweise}\label{labelundref}
```

Listing 4.16: Beispiel für ein Label

Um auf ein Label zu verweisen, z. B. Listing 4.16 kann der Befehl `~\ref{}` genutzt werden. Innerhalb der geschweiften Klammern wird dann das passende Label hinterlegt, dies kann dann so aussehen:

```
1 Um auf ein Label zu verweisen, z. B. Listing~\ref{beispiellabel} kann der
  ↪  Befehl...
```

Listing 4.17: Beispiel für eine Referenz

5 Literaturverwaltung und zitieren

Wenn wir darüber sprechen, Hausarbeiten zu schreiben, dann brauchen wir in jedem Fall eine gute Literaturverwaltung. LaTeX bietet uns hierfür mit BibTeX ein sehr einfaches und verbreitetes Tool. Wir können es nutzen, um im Text zu zitieren bzw. zu paraphrasieren und um uns dann ein Literaturverzeichnis erstellen zu lassen. Genaugenommen arbeiten wir in diesem Template nicht mit BibTeX, sondern mit BibLaTeX, welches einen besseren Umgang mit Umlauten und Webseiten bietet. Ich verwende den Begriff BibTeX jedoch an dieser Stelle weiter.
Damit das Arbeiten mit BibTeX möglich ist, brauchen wir unsere Literatur als BibTeX Einträge in unserem Projekt. Ich habe hierfür ein Verzeichnis (bibtex) angelegt, welches eine

Datei (hauptdatei.bib) enthält. In dieser Datei können wir unsere Literatureinträge hinterlegen. Glücklicherweise bietet quasi jede Bibliotheksseite einen entsprechenden Export in BibTeX an:

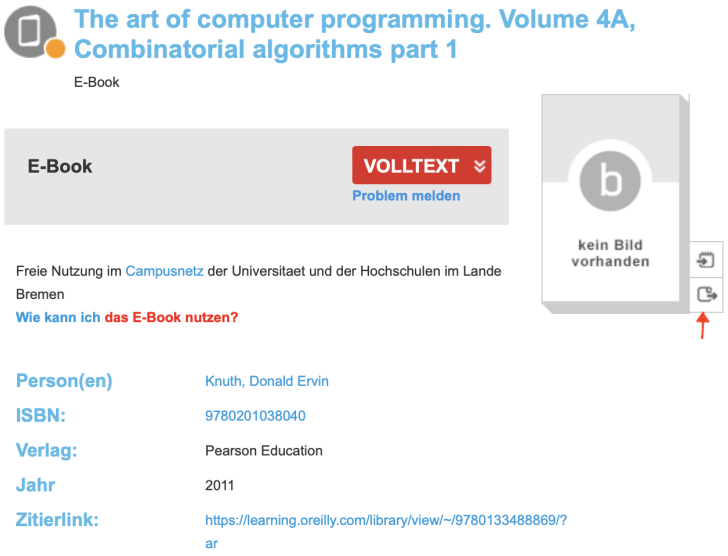

Abbildung 5.1: BibTeX Einträge exportieren

Die Möglichkeit sich zu Artikeln, Büchern, Sammelbänden etc. die passenden BibTeX Einträge generieren zu lassen, wird auf den entsprechenden Seiten quasi flächendeckend angeboten. Resultat eines solchen Exports ist ein einzelner BibTeX Eintrag, dieser kann dann in der `hauptdatei.bib` hinterlegt werden.

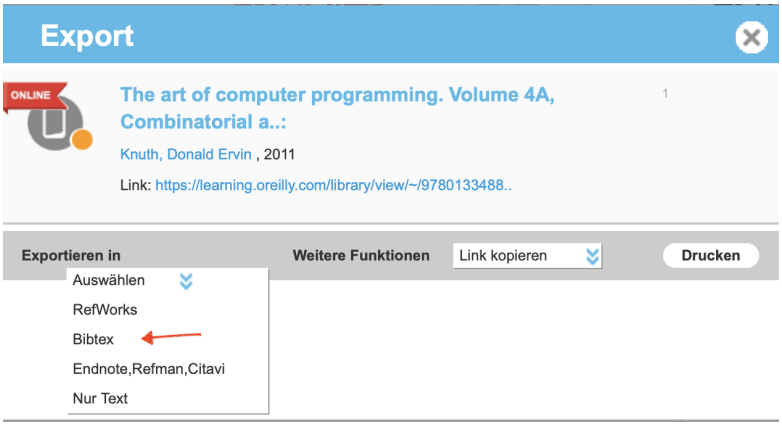

Abbildung 5.2: BibTeX Einträge exportieren (Formatauswahl)

```
@book{Thea2011,
title = {The art of computer programming. Volume 4A, Combinatorial algorithms part
↪  1},
author = {Donald Ervin Knuth},
publisher = {Pearson Education},
year = {2011},
note = {1 volume},
ISBN = {9780201038040},
URL = {https://learning.oreilly.com/library/view/~/9780133488869/?ar},
}
```

Listing 5.1: Beispiel BibTeX Eintrag

Im Laufe einer Hausarbeit wird unsere persönliche Bibliothek eine Vielzahl solcher Einträge enthalten. Die Einträge enthalten jeweils einen Typ, im obigen Beispiel `book` und ein Kürzel, in diesem Fall `Thea2011`. Es existieren viele verschiedene Typen von Einträgen, in den meisten Fällen werden wir unsere Einträge, wie gezeigt, exportieren können, es ist natürlich auch möglich, eigene Einträge zu erstellen. Hauptsächlich tue ich dies, wenn es sich um Webseiten oder ähnliche Onlinequellen handelt. Ein solcher Eintrag lässt sich gemäß nachstehendem Format erstellen:

```
  @online{flignitz,
  title = {How-to LaTeX},
  url = {https://informatik.hs-bremerhaven.de/flignitz/latex.html},
  urldate = {2023-04-18},
  Author = {Fabian Lignitz},
  Language = {de},
  Month = {04},
  Year = {2023},
  }
```

Listing 5.2: Beispiel BibTeX Eintrag (Onlinequelle)

Soll eine Literaturangabe nach einem Zitat oder Paraphrase angefügt werden, ist dies durch das Kommando `~\autocite[]{}` möglich. Zur Veranschaulichung sollen uns einige kleine Beispiele dienen:

„In general, each prime implicant corresponds in this way to a maximal subcube that stays within the set of points that make f true [1].“

> When you hear about this problem for the first time, you might be tempted to ask a question of your own in return: „What? Are you serious that computer scientists still haven't figured out how to do such a simple thing?“ [1, S.432]

Gemäß Lignitz [2] biete BibLaTeX einen besseren Umgang mit Umlauten und Onlinequellen, als dies mit BibTeX möglich wäre.

Wie in den Beispielen [3] zu sehen, kann an jeder beliebigen Stelle zitiert werden, zusätzlich kann in den eckigen Klammern eine Seitenzahl angegeben werden, dies ist jedoch nicht nötig. Zu beachten gilt, dass eingerückte Zitate keine Anführungsstriche erhalten, jedoch in diesem Fall in der Quelle Anführungsstriche gesetzt wurden und diese beibehalten werden müssen. Paraphrasen erhalten in keinem Fall Anführungszeichen. In LaTeX sehen die Beispiele wie folgt aus:

```
\glqq In general, each prime implicant corresponds in this way to a maximal
subcube that stays within the set of points that make f
true~\autocite[]{KnutThea2009}.\grqq{}

\begin{quote}
  \footnotesize{
  When you hear about this problem for the first time, you might be tempted
  to ask a question of your own in return: \glqq What? Are you serious that computer
  scientists still haven't figured out how to do such a simple
  thing?\grqq{}~\autocite[S.432]{KnutThea2009}
  }
\end{quote}

Gemäß Lignitz~\autocite[]{flignitz} biete BibLaTeX einen besseren Umgang mit
Umlauten und Onlinequellen, als dies mit BibTeX möglich wäre.
```

Listing 5.3: Beispiele zitieren

Sobald wir in unserem Fließtext zitieren, wird entsprechend des gewählten Zitierstils auch das Literaturverzeichnis gebaut. Dies findet ihr am Ende des Hauptteils als erstes Verzeichnis. Durch die Verwendung von BibTeX brauchen wir uns also weder um das Anlegen von Literaturverzeichnis, noch um die Nummerierung oder Reihenfolge von Literaturangaben zu kümmern. Es reicht lediglich mittels des `~\autocite[]{}` Kommandos und des richtigen Kürzels im Fließtext eine Literaturangabe zu setzen.

6 Das Template herunterladen und bauen

Mittels all dieser Kommandos und Umgebungen ist es uns nun möglich, eine vollständige Hausarbeit zu schreiben. Dafür müsst ihr das LaTeX Projekt lediglich herunterladen und könnt quasi sofort loslegen. Wie bereits erwähnt, liegt in diesem Projekt der Inhalt dieser Anleitung. Es ist euch also ohne weiteres möglich, den gesamten Inhalt auch nochmal im Quelltext nachzulesen. In der Infrastruktur der Informatik kann das Projekt ohne Probleme gebaut werden und es sind keine weiteren Schritte notwendig. Auch mittels Overleaf sollte es ohne Probleme möglich sein, das Projekt zu importieren und zu bauen. Lediglich lokal auf eurem Rechner müsst ihr einige Programme installieren, zum einen braucht ihr LaTeX für euer Betriebssystem, zum anderen wird eine Form von Editor benötigt. Auf macOS und Linux steht euch der Vim zur Verfügung, auf Windows empfehle ich die Installation der WSL, da LaTeX unter Windows nicht unbedingt angenehm zu nutzen ist und mein Projekt auch nicht ohne weiteres unter Windows gebaut werden kann. Zusätzlich wird noch das Programm `pygments` benötigt.
Steht die passende Umgebung zur Verfügung, könnt ihr mein Projekt entweder über den Browser oder noch einfacher über die Kommandozeile herunterladen. Die passenden Befehle und URL findet ihr auf meiner Webseite.
Um das gesamte Dokument zu bauen (kompilieren) reicht es, das hinterlegte Skript auszuführen. Grundsätzlich läuft das Kompilieren mehrfach durch, dies ist unter LaTeX üblich. Leider bietet der Compiler nicht unbedingt sehr aussagekräftige Fehlermeldungen, es gilt die Ausgabe genau zu lesen. Ebenfalls kann der Compiler auch mit fehlerhaftem Code oftmals ein PDF produzieren, dies enthält dann jedoch nicht zwingend den gewünschten Inhalt. Ihr solltet entsprechend darauf achten, ob der Prozess tatsächlich sauber durchläuft. Dies wird auch am Ende des Kompilierens ausgegeben, der Fehlercode `1` ist dabei das eindeutigste Indiz für einen Fehler.

Literaturverzeichnis

[1] D. E. Knuth, *The art of computer programming / Donald E. Knuth: Combinatorial algorithms Fasc. 1: Bitwise tricks & techniques : binary decision diagrams*. Upper Saddle River, NJ [u.a.]: Addison-Wesley, 2009, ISBN: 0321580508.

[2] F. Lignitz. „How-to LaTeX.“ de. (Apr. 2023), Adresse: https://informatik.hs-bremerhaven.de/flignitz/latex.html (besucht am 18. 04. 2023).

[3] N. Stephenson, „In the beginning was the command line,“ 2004. Adresse: https://wiki.chadnet.org/files/in-the-beginning-was-the-command-line-by-neal-stephenson.pdf (besucht am 17. 01. 2023).

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Anhang

I Die Poster

Passend zu diesem Template existieren noch zwei weitere Vorlagen für Poster. Diese unterscheiden sich lediglich durch die Ausrichtung der Seite und sind ansonsten identisch. Der Aufbau der Dateien und Projektstruktur ist weitestgehend identisch zu dieser Vorlage, es gilt lediglich auf einige kleine Besonderheiten zu verweisen.
In der sog. `multicols` Umgebung ist es nicht möglich, mit Floatumgebungen zu arbeiten, dies ist für fast alle Beispiele aus dieser Anleitung irrelevant, lediglich Grafiken müssen anders eingebunden werden. Dazu finden sich jedoch Beispiele in den Vorlagen. Ebenso ist ein Teil der Präambel in eine separate Datei ausgelagert, in dieser sollten auch nicht ohne weiteres Änderungen vorgenommen werden. Wie in diesem Template, müssen innerhalb der Präambel Zitierstil, sowie Autor:innen und Titel gesetzt werden.
Die Vorlagen können auf meiner Webseite sowohl angeschaut, als auch heruntergeladen werden, alle Hinweise zum Kompilieren gelten für die Poster ebenso. Ergebnis dieser Vorlage ist z. B. folgendes Poster:

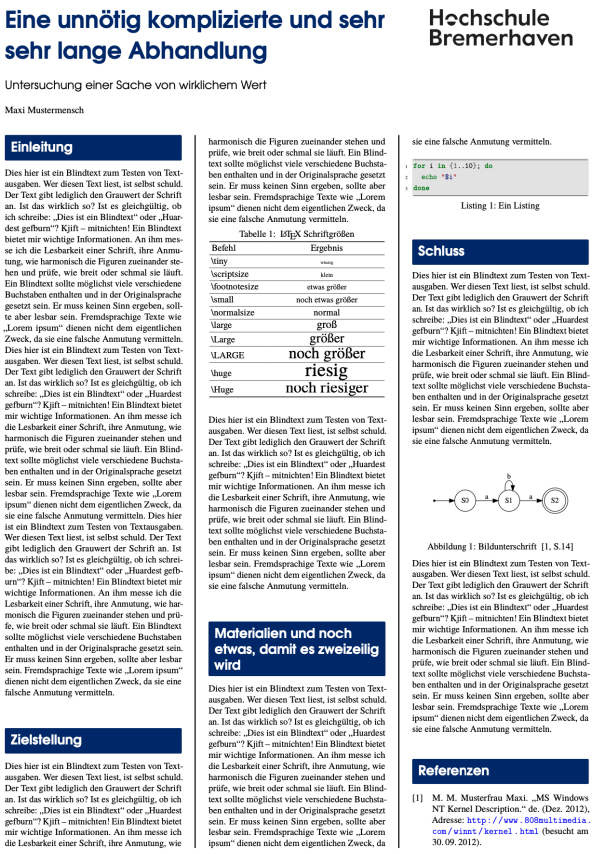

Abbildung I.1: Poster

II Beamer

LaTeX bietet nicht nur die Möglichkeit schriftliche Ausarbeitungen in Form von Hausarbeiten zu erstellen, sondern mit Beamer eine Alternative zu PowerPoint oder Keynote. Das Layout ist in vielen Fällen schlicht, was nicht negativ gemeint ist und erinnert im Design eher an die frühen 2000er. Durch einige Anpassungen ist es dennoch möglich ein schlichtes, aber funktionales Layout zu entwerfen, welches sich gut für Präsentationen eignet.
Eine solche Vorlage findet sich analog zu den Poster Vorlagen und erfordert wenig Umgewöhnung. Es können Tabellen und Abbildungen, sowie Referenzen wie gewohnt genutzt werden. Lediglich die Listings erfordern eine kleine Umgewöhnung, ebenso wie das Arbeiten mit der *column* Umgebung.

```
\section{Frame Example with table}

\begin{frame}{Frame Example with table}
  \begin{columns}

  \begin{column}{0.4\textwidth}
    This is some text in the second frame~\autocite{donovan2015go}.
    This is some text in the second~\autocite{kane2018docker}.
  \end{column}

  \begin{column}{0.59\textwidth}
    \scriptsize{
      \begin{talltblr}[caption={\LaTeX~Sonderzeichen}]{colspec={X[1,m] X[0.5,c,m]},
      ↪  rowhead=1, width=0.8\textwidth}\toprule
      Befehl                          & Ergebnis         \\ \midrule
      \textbackslash\&                & \&               \\ \cmidrule{1-2}
      \textbackslash\%                & \%               \\ \cmidrule{1-2}
      \textbackslash\$                & \$               \\ \cmidrule{1-2}
      \textbackslash\#                & \#               \\ \cmidrule{1-2}
      \textbackslash\{                & \{               \\ \cmidrule{1-2}
      \textbackslash\}                & \}               \\ \cmidrule{1-2}
      \end{talltblr}}
  \end{column}

  \end{columns}
\end{frame}
```

Listing II.1: Beamer Frame

Wie zu sehen, wird in Beamer ebenfalls mit `sections` gearbeitet. Um einen neuen Slide zu erstellen wird `\begin{frame}` genutzt. Die `columns` Umgebung ermöglicht das Unterteilen des Frames in mehrere Spalten, welche daraufhin als einzelne `column` definiert werden. Dabei kann die Textbreite angegeben werden.
Listings müssen über eine besondere Umgebung definiert und eingebunden werden, da Beamer ansonsten nicht mit `minted` umgehen kann.

```
\defverbatim[colored]\CodeOne{
  \begin{minted}{python}
    import numpy as np
    import pylab as pl

    def f_x(x):
      return np.exp(x)+x**2-5*x

    def approx_f(x):
      return 1 -4*x +3./2*x**2

    xvals = np.arange(-4,4,0.1)
    fx_vals = [f_x(x) for x in xvals]
    approx_vals = [approx_f(x) for x in xvals]

    pl.plot(xvals,fx_vals)
    pl.plot(xvals,approx_vals)

    pl.show()
\end{minted}
\captionof{listing}{Example Code}
}

\section{Code Example}
\begin{frame}{Code Example}
  Some Text on first Frame~\autocite{gregg2013systems}
  \CodeOne
  \vspace{0.5cm}
\end{frame}
```

Listing II.2: Beamer Frame mit Listing

Wie im Listing gezeigt, wird das Listing außerhalb des Frames definiert und in Zeile 27 eingefügt. `CodeOne` ist dabei der zugewiesene Name unter dem das Listing eingefügt werden kann. Die Definition erfolgt in Zeile 1, der Name ist frei wählbar. Das Listing so einzubinden ist nicht optimal, jedoch bleibt aktuell keine andere Möglichkeit, wenn mit `minted` gearbeitet wird.
Das Inhaltsverzeichnis und die Referenzen werden automatisch erstellt, alle Einstellungen können natürlich innerhalb der Vorlage angepasst werden.
Obiges Listing würde mit der Vorlage folgendermaßen aussehen. Wie immer finden sich alle Beispiele auch in den jeweiligen Vorlagen.

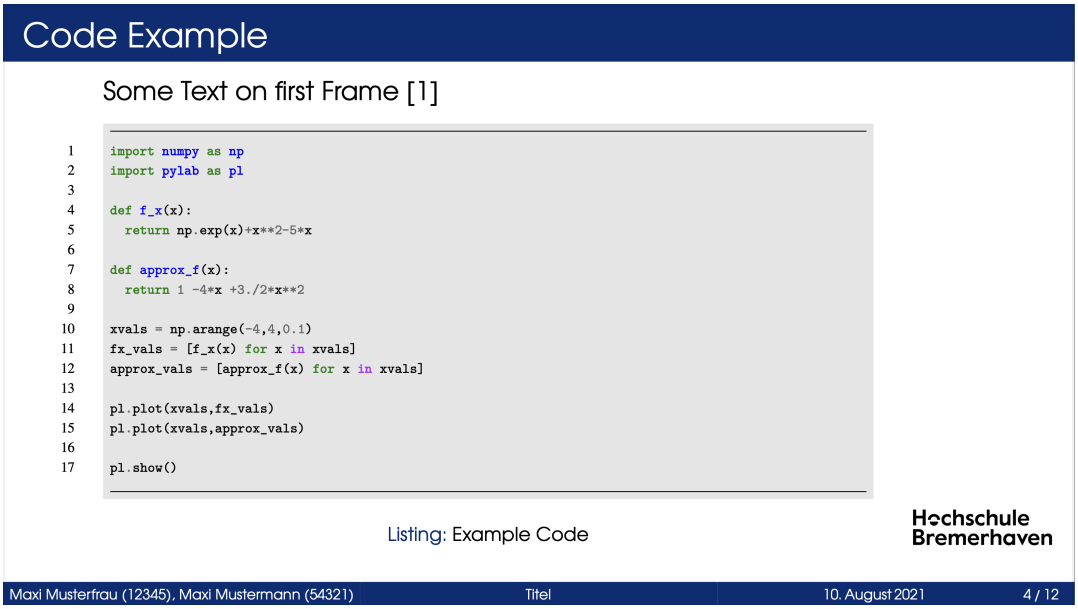

Abbildung II.1: Frame mit Listing

III Quickbuild

Wird ein Projekt mit der Zeit relativ groß, verfügt also über eine Vielzahl von Listings, Tabellen, Abbildungen und Text, dann wird für das Kompilieren eine nicht unerhebliche Zeit benötigt. Um das ganze Dokument, inkl. aller Referenzen und Abkürzungen etc. zu bauen, ist dies auch nötig und lässt sich nicht vermeiden. Oft benötigt man jedoch nur einen kurzen Blick auf Layout oder den Umfang des bisher geschriebenen und möchte nicht das Dokument in seiner Gesamtheit kompilieren.
Für diese Anforderungen gibt es sog. `subfiles`, diese ermöglichen es, nur einen kleinen Teil des Dokumentes zu kompilieren und als PDF zu betrachten. Dabei kommt es zu gewissen Einschränkungen, so werden Seitenzahlen, Referenzen und Abkürzungen nicht korrekt dargestellt, dennoch erhält man einen guten Blick auf das Layout. Um mit `subfiles` zu arbeiten, muss einiges beachtet werden. So muss statt des `\input{}` das Kommando `\subfile{}` genutzt werden. Darüber hinaus müssen alle `.tex`-Dateien mit einer eigenen `documentclass` ausgestattet werden. In der `hauptdatei.tex` sieht dies dann folgendermaßen aus.

```
\subfile{src/einleitung.tex}
\subfile{src/struktur.tex}
\subfile{src/zurhauptdatei.tex}
```

Listing III.1: Einbinden von subfiles

Jedes Subfile muss dann mit der passenden `documentclass` beginnen.

```
\documentclass[../hauptdatei.tex]{subfiles}
\begin{document}
\section{Beispiel}

Hier könnte sinnvoller Inhalt stehen.

\end{document}
```

Listing III.2: Erstellen von subfiles

Durch die eigene `documentclass`, welche wiederum auf die `hauptdatei.tex` verweist und die Angabe von `\begin` bzw. `\end{document}` kann die entsprechende `tex`-Datei separat kompiliert werden. Dafür kann das Skript `quickbild.sh` genutzt werden, welches als Argument den Namen der zu kompilierenden Datei erwartet und ein PDF erzeugt.
Abbildungen, Listings, Tabellen können wie gewohnt genutzt werden. Wird `buildlualatex.sh` ausgeführt, baut nach wie vor das gesamte Dokument inkl. aller Subfiles und Referenzen.

Selbstständigkeitserklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit habe ich mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegt.

Bremerhaven, den 28. November 2023 Unterschrift: