Kompaktkurs zu LaTeX 2_ε

Dirk Nitschke   Wolfgang Ripken

2. März 1998

Inhaltsverzeichnis

Tabellenverzeichnis

1 Das Konzept von LaTeX

1.1 Wer bin ich?

LaTeX (sprich: lah-tech) ist der Name eines Systems zum Setzen von Dokumenten. Es handelt sich dabei um eine von Leslie Lamport initiierte Erweiterung des Satzsystems TeX (sprich: tech), welches von Donald E. Knuth entwickelt wurde. Die erste weit verbreitete Version von LaTeX trug die Nummer 2.09. In der folgenden Zeit wurden viele Verbesserungen an dieser Version vorgenommen – leider verlief dies unkoordiniert, so daß es zu einem Durcheinander von Versionen kam. Abhilfe schafft die aktuelle und offizielle Version LaTeX 2_ε. In diese Version wurden unter anderem viele Verbesserungen zur Einbindung von Grafiken, farbigen Textteilen und zur Behandlung von unterschiedlichen Schrifttypen eingearbeitet. Wenn wir im folgenden über LaTeX sprechen, so meinen wir – solange nicht anderes erwähnt wird – stets LaTeX 2_ε.

1.2 Wozu LaTeX? oder: Textsatz damals und heute

Bevor Computer Einzug in unser Leben hielten, reichte ein Autor ein hand- oder maschinengeschriebenes Manuskript bei einem Verlag ein. *Dort* wurde über die *typographische Gestaltung* des Textes entschieden: Zeilenlänge, Abstände zwischen Absätzen, die zu verwendenden Schrifttypen und vieles mehr mußten festgelegt werden, um ein ansprechendes und ausgeglichenes Äußeres zu erzielen. Hieraus ergaben sich Anweisungen für den Setzer, mit denen er entschied, an welcher Stelle der Seite die Zeichen und Worte des Autors erschienen.

Heute werden Manuskripte mit dem Computer erstellt und dann an den Verlag geschickt. Aus Kostengründen vervielfältigt der Verlag häufig den Text nur noch, bindet und verschickt ihn. Die typographische Gestaltung, bei der es sich um ein Handwerk handelt, welches *erlernt werden muß,* liegt somit beim Autor. Da von uns nur wenige die Möglichkeit haben werden, dieses Handwerk zu erlernen, liegt es nahe, die Aufgabe dem Computer zu übertragen und sich selbst ganz auf den *Inhalt* und die *Struktur* des Textes zu konzentrieren. Die Verwendung von LaTeX bewerkstelligt genau diese Arbeitsteilung: LaTeX übernimmt die *typographische Gestaltung*, TeX erledigt den *Textsatz* und Sie sind für die *Struktur* und den *Inhalt* des Textes zuständig.

Menschliche Gestalter wissen, wovon ein Manuskript handelt. Sie *erkennen* an den Formulierungen, ob es sich beim gerade zu bearbeitenden Textteil um eine Überschrift, ein Zitat, eine mathematische Formel oder eine Auflistung handelt. Diese Informationen fließen in den Textsatz mit ein. Ein

Computerprogramm kann Ihren Text nicht verstehen und ohne Hilfe Ihrerseits nicht entscheiden, welche logische Bedeutung ein Textteil hat. Somit müssen Sie LaTeX die *logische Struktur* des Textes durch spezielle *Befehle* mitteilen. Diese Methode des Textsatzes nennt man *logisches Design.*

„Das kann … doch auch alles – und das ist einfacher zu bedienen!“ werden Sie sagen. Dabei handelt es sich bei … wahrscheinlich um ein sogenanntes WYSIWYG-Programm, welches Ihnen „what you see is what you get“ („man bekommt, was man sieht“ – für mich eher „man bekommt, was man verdient“) verspricht. Bei solchen Programmen haben Sie meist die Möglichkeit, Ihren Text in beliebiger Größe und Schriftart an einer beliebigen Stelle auf der Seite zu plazieren und hin und her zu schieben. Dies hat aber nichts mit der logischen Struktur des Textes zu tun, sondern mit dem eigentlichen Textsatz. Sie teilen dem Programm *nicht* mit, wie der Text aufgebaut ist, sondern wie er *aussehen* soll. Dies bezeichnet man auch als *visuelles Design.* Die Verwendung von logischem statt visuellem Design ist es, was LaTeX von WYSIWYG-Programmen unterscheidet.

1.3 Logisches vs. visuelles Design

Visuelles Design und damit WYSIWYG-Programme eignen sich durch ihre leichte Handhabbarkeit gut zum Schreiben von *kurzen* Texten, da man mit ihnen *mal schnell* etwas eingeben kann. Bei umfangreicheren Texten und insbesondere wissenschaftliche Arbeiten kann *mal schnell* leicht in erheblich mehr Arbeit ausarten. Dazu ein Beispiel: Nehmen wir an, in Ihrem Text kommt der (mathematische) Ausdruck (A, B) vor. Bei einem WYSIWYG-Programm geben Sie einfach `(A,B)` ein. In LaTeX könnten Sie das auch so eingeben, jedoch würden wir dann *visuelles* Design statt *logischem* Design betreiben. Wir müssen daher LaTeX mitteilen, daß es sich um einen mathematischen Textteil handelt. Dies wäre zum Beispiel durch die Eingabe von `$(A,B)$` zu bewerkstelligen. Wir haben die logische Struktur aber noch nicht vollständig freigelegt: Dieser Ausdruck hat auch eine besondere mathematische Bedeutung und bezeichnet häufig ein *Skalarprodukt.* Fortgeschrittene LaTeX-Benutzer werden daher einen Befehl definieren, der dies wiedergibt. Ein solcher Befehl könnte `\skp` heißen und die Eingabe wäre dann `$\skp{A}{B}$`. Taucht jetzt in Ihrem Text ein *weiteres* Skalarprodukt (C, D) auf, so können Sie den Befehl `\skp` erneut verwenden und `$\skp{C}{D}$` eingeben.

Wozu das alles? Warum soll ich nicht einfach `(A,B)` und `(C,D)` eingeben? Diese Frage zielt auf den Sinn logischen Designs. Nehmen wir das Beispiel von eben. Sollten Sie sich – nachdem Sie bereits einige Seiten Ihres Textes eingegeben haben – dazu entschließen, ein Skalarprodukt *anders* zu bezeichnen, etwa $\langle A, B\rangle$ statt (A, B), so brauchen Sie bei *logischem* Design nur an *einer* Stelle Ihrer Eingaben eine Änderung vornehmen, nämlich in der Definition des Befehls `\skp`. Bei *visuellem* Design müssen Sie bei jedem Auftreten von `(A,B)`, `(C,D)` und *allen weiteren* irgendwo eingegebenen Skalarprodukten die entsprechenden Änderungen von Hand vornehmen und jedesmal entscheiden, ob eine Korrektur nötig ist.

Ein weiterer Vorteil liegt darin, daß sich viele arbeitsintensive und fehlerträchtige Aufgaben durch logisches Design vereinfachen – d. h. dem Computer übertragen – lassen. Dazu zählt zum Beispiel die automatische Numerierung und Referenzierung von Textabschnitten, Fußnoten, mathematischen Formeln, Tabellen und Literaturangaben sowie das Erstellen von Inhalts- und Schlagwortverzeichnissen.

Das ganze hat aber auch Nachteile: Eine umfangreichere Änderung an den LaTeX bekannten Formatierungsregeln ist nicht ganz einfach und sollte erfahrenen Benutzern vorbehalten bleiben.

2 Eingabe von Texten

2.1 Eingabedatei und Ausgabedateien

LaTeX entnimmt den zu setzenden Text und alle speziellen Befehle zur logischen Struktur des Textes einer *Eingabedatei,* die man mit einem Editor seiner Wahl erstellen kann. Im allgemeinen besteht der *Dateiname* einer solchen Eingabedatei aus zwei Teilen, die durch einen Punkt voneinander getrennt werden. Der vordere Teil wird als *Grundname* und der hintere als *Erweiterung* bezeichnet. Der Grundname Ihrer Eingabedateien kann beliebig sein (bis auf die Beschränkungen, die Ihr Computersystem Ihnen vorgibt: Unter DOS darf der Grundname nur 8 Zeichen lang sein, auf dem Macintosh jedoch 31 Zeichen lang). Die Erweiterung sollte `tex` lauten (auf Systemen, die Groß- und Kleinschreibung in Dateinamen unterscheiden, soll `tex` klein geschrieben werden).

LaTeX erzeugt mehrere Ausgabedateien, deren Grundname so lautet wie der der Eingabedatei, die Sie von LaTeX bearbeiten lassen. Die wichtigste dieser Dateien ist diejenige mit der Erweiterung `dvi`. Diese Datei benötigen Sie zum Betrachten und Ausdrucken des gesetzten Textes.

2.2 Aufbau einer Eingabedatei

Eine Eingabedatei besteht aus zwei Teilen, der *Präambel* und dem *Textteil.*

In der Präambel teilt man LaTeX als erstes mit, welche *Dokumentenklasse* – d. h. welches Layout – für das folgende Dokument verwendet werden soll. Dies geschieht mit dem Befehl

`\documentclass[`⟨*Optionen*⟩`]{`⟨*Klassenname*⟩`}`.

Einige der standardmäßig zu LaTeX 2_ε gehörenden Klassen finden Sie mit einer kurzen Beschreibung in Tabelle 1.

Erweiterungen von LaTeX – wie zum Beispiel spezielle Unterstützung für deutschsprachige Texte oder zusätzliche Strukturen für mathematischen Formelsatz – werden in *Paketen* zusammengefaßt. Mit dem Befehl

`\usepackage[`⟨*Optionen*⟩`]{`⟨*Paketnamen*⟩`}`

kann man in der Präambel die Vereinbarungen aus solchen Paketen verwenden. Eine kleine Auswahl an Paketen zeigt Tabelle 2. Außerdem können in der Präambel kleinere Änderungen an den Layoutvorgaben oder eigene Befehle vereinbart werden.

Der Textteil eines Dokumentes beginnt *immer* mit dem Befehl

Tabelle 1: Dokumentklassen

article Wird für Artikel in wissenschaftlichen Zeitschriften, Vorträge, Praktikumsarbeiten, Seminararbeiten, kürzere Berichte, Anträge, Gutachten, Programmbeschreibungen, Einladungen u. s. w. verwendet.

report Für längere Berichte, die aus mehreren Kapiteln bestehen, Diplomarbeiten, Dissertationen, Skripte u. s. w.

book Wird für Bücher verwendet.

slides Speziell zur Erstellung von Folien.

letter Eine Dokumentenklasse für Briefe nach amerikanischem Muster.

proc Geeignet für *Proceedings* (Konferenzbände).

Tabelle 2: Pakete

german Spezielle Unterstützung für das Schreiben von deutschsprachigen Texten. So wird unter anderem die Eingabe von Umlauten vereinfacht.

amsmath Ergänzungen für den Satz mathematischer Texte.

amsthm Befehle zur Definition „theoremartiger“ Strukturen.

```
\begin{document}
```

und endet mit dem Befehl

```
\end{document}.
```

Zwischen diesen Befehlen steht der Text. Dieser wird von TeX automatisch in Zeilen gebrochen, wobei Worte gegebenenfalls getrennt werden.

Alles, was Sie hinter `\end{document}` eingeben, wird von LaTeX ignoriert.

2.3 Zulässige Zeichen

Eine Eingabedatei darf leider nicht alle Zeichen enthalten, die Sie auf Ihrer Tastatur finden. Zulässige Zeichen sind:

Groß- und Kleinbuchstaben: `A B` ... `Z a b` ... `z`
Ziffern: `0 1` ... `9`
Satzzeichen: `. : ; ? ! ‘ ’ ( ) [ ] - / * @`
Mathematische Zeichen: `+ = | < >`
Sonderzeichen: `# % $ & _ { } ~ ^ \ "`
Unsichtbare Zeichen: ⟨*Leertaste*⟩ ⟨*Tabulatortaste*⟩ ⟨*Zeilenendetaste*⟩

Hieraus ergibt sich gleich eine (für deutschsprachige Benutzer unerfreuliche) Einschränkung:

Die deutschen Umlaute sind *nicht* zugelassen!

Welche Bedeutung haben diese Zeichen in einer Eingabedatei?

Groß- und Kleinbuchstaben, Ziffern und Satzzeichen interpretiert LaTeX genau so, wie man sie eingibt. Die Eingabe

```
Heute regnet es.
```

erzeugt somit die Ausgabe

Heute regnet es.

Die fünf Zeichen

```
+ = | < >
```

sind für mathematische Formeln gedacht. + und = können aber auch im normalen Text verwendet werden (Auf den Unterschied zwischen „normalem“ und mathematischem Text wird in Abschnitt 10 eingegangen).

Die zehn Sonderzeichen

```
# % $ & _ { } ~ ^ \
```

haben für LaTeX eine besondere Bedeutung. Wenn man eines dieser Zeichen im laufenden Text erzeugen möchte – also zum Beispiel „Dieses Skript kostet 95 $“ schreiben will –, so muß man dies LaTeX durch *Befehle* mitteilen. Das elfte Sonderzeichen `"` hat eine besondere Bedeutung, wenn das `german`-Paket benutzt wird.

Darüber hinaus sind noch *unsichtbare* Zeichen erlaubt: Leerzeichen (durch Drücken der Leertaste), Tabulatoren (Tabulatortaste) und die Zeichen für das Zeilenende (Return oder Enter). All diese unsichtbaren Zeichen werden von TeX gleich behandelt. Mehrere Leerzeichen bewirken dasselbe wie *ein* Leerzeichen. Eine Leerzeile wird von TeX als das Ende eines Absatzes verstanden. Mehrere Leerzeilen wirken wie eine Leerzeile. Das bedeutet, daß es *nicht* möglich ist, durch mehrere Leerzeichen bzw. Leerzeilen den Abstand zwischen zwei Worten bzw. Absätzen zu vergrößern (das hatten Sie doch wohl nicht etwa vor?).

2.4 Befehle und Umgebungen

Es war eben schon von *Befehlen* die Rede. Sie bestehen entweder aus einem einzelnen Zeichen wie `~`, einem *Backslash* `\` gefolgt von genau *einem* nichtalphabetischen Zeichen (also keinem Groß- oder Kleinbuchstaben) wie zum Beispiel `\"` oder einem `\` gefolgt von einer Folge von Groß- und Kleinbuchstaben wie `\LaTeX`. Leerzeichen und ein Zeilenende nach den letztgenannten Befehlen werden ignoriert. Um nach einem solchen Befehl einen Wortzwischenraum zu erzeugen, muß man einen weiteren Befehl eingeben: `\␣`, also einen Backslash gefolgt von einem Leerzeichen. Einige Befehle haben eine *Sternform,* die man erhält, indem man direkt nach dem Befehl das Zeichen `*` eingibt.

Viele Befehle haben eines oder mehrere *Argumente,* die man zwingend angeben muß und in geschweifte Klammern `{` und `}` einfaßt. Es gibt auch Befehle, die *optionale Argumente* haben. Solche *können* angegeben werden, müssen es aber nicht. Falls sie angegeben werden, so werden sie von `[` und `]` begrenzt.

Bei *Umgebungen* handelt es sich um ein Paar

`\begin{`⟨*Name*⟩`}` ... `\end{`⟨*Name*⟩`}`

von Befehlen. Eine solche Umgebung haben wir bereits kennengelernt: die `document`-Umgebung, die den Textteil einschließt.

2.5 Sätze und Absätze

TeX ignoriert die Art und Weise, wie die Eingabedatei formatiert ist und achtet nur auf die logischen Konzepte Wortende, Satzende und Absatzende. Dadurch gestaltet sich die Eingabe von einfachen Sätzen und Absätzen recht einfach, da man den Text wie auf der Schreibmaschine schreibt. Der Zeilenumbruch und die Trennung von Worten wird allerdings von TeX übernommen.

```
Mit  Leerzeichen kennzeichnet  man
das Ende eines Wortes oder Satzes.
Dabei ist es unerheblich, wieviele
Leerzeichen           man eingibt.
1 ist so gut wie              15.


Eine oder mehrere Leerzeilen zeigen
das Ende eines Absatzes an.
```

Mit Leerzeichen kennzeichnet man das Ende eines Wortes oder Satzes. Dabei ist es unerheblich, wieviele Leerzeichen man eingibt. 1 ist so gut wie 15.

Eine oder mehrere Leerzeilen zeigen das Ende eines Absatzes an.

2.6 Eingabe besonderer und unzulässiger Zeichen

2.6.1 Besondere Zeichen

Eben wurde erwähnt, daß die Zeichen

```
# % $ & _ { } ~ ^ \ "
```

für LaTeX eine besondere Bedeutung haben und nicht zur Erzeugung der entsprechenden Zeichen benutzt werden können. Da zumindest die ersten sieben dieser Zeichen ab und zu in gewöhnlichen Texten auftauchen, kann man sie durch Voranstellen des `\` erzeugen:

`\# \% \$ \& \_ \{ \}`   # % $ & _ { }

(Wenn Sie die übrigen vier Zeichen `~`, `^`, `\`, `"` wirklich erzeugen wollen, *fragen Sie jemanden, der sich damit auskennt.*)

Tabelle 3: Akzente

Eingabe	Ausgabe	Eingabe	Ausgabe
\`{o}	ò	\u{o}	ŏ
\'{o}	ó	\v{o}	ǒ
\^{o}	ô	\H{o}	ő
\"{o}	ö	\t{oo}	o͡o
\~{o}	õ	\c{o}	o̧
\={o}	ō	\d{o}	ọ
\.{o}	ȯ	\b{o}	o̲
\r{o}	o̊		

Tabelle 4: Sprachspezifische Zeichen

Eingabe	Ausgabe	Eingabe	Ausgabe
\oe	œ	\OE	Œ
\o	ø	\O	Ø
\ae	æ	\AE	Æ
\l	ł	\L	Ł
\aa	å	\AA	Å
?`	¿	!`	¡
\ss	ß		

2.6.2 Akzente und andere Spezialitäten

Die deutschen Umlaute sind in der Eingabedatei nicht zulässig. Dasselbe gilt für akzentuierte Zeichen anderer Sprachen. Diese Zeichen lassen sich durch spezielle Befehle ausgeben, die in Tabelle 3 angegeben sind. Die Tabelle zeigt, wie man die entsprechenden Akzente an dem Zeichen `o` anbringt. Sie können `o` aber durch jeden anderen Buchstaben ersetzen. Bei `i` und `j` muß man allerdings beachten, daß der Punkt verschwindet, wenn sie einen Akzent erhalten sollen. Auch hierfür ist TEX gerüstet und erzeugt durch die Befehle `\i` und `\j` die punktlosen Varianten ı und ȷ.

`\'{E}l est\'{a} aqu\'{\i}.`	Él está aquí.

Neben Akzenten gibt es noch Zeichen, die sich nur in bestimmten Sprachen finden. Tabelle 4 zeigt, mit welchem Befehl man welches Zeichen erzeugen kann. Leider muß man auch hier etwas beachten:

All diese Befehle können nur im Textmodus verwendet werden!

Erfreulicher verhalten sich die folgenden Befehle, die man sowohl in gewöhnlichem Text als auch im Mathematikmodus benutzen kann:

`\dag`	†	`\S`	§	`\copyright`	©
`\ddag`	‡	`\P`	¶	`\pounds`	£

2.6.3 Anführungszeichen und Bindestriche

Weitere besondere Zeichen sind Anführungszeichen und Bindestriche. Da TeX ein Programm zum *Textsatz* ist und nicht *Schreibmaschinensatz,* muß man bei diesen Zeichen etwas aufpassen. Insbesondere gibt es Unterschiede zwischen deutsch- und englischsprachigen Texten.

Im Maschinensatz verwendet man für Anführungszeichen entweder das doppelte `"` oder das einfache Anführungszeichen `'`, welches auch als Apostroph Verwendung findet. Im deutschen Schriftsatz werden vornehmlich die Zeichen „ und “ benutzt.

Der Buchsatz kennt auch drei verschiedene Bindestriche. Ihre Verwendung ist ebenfalls sprachabhängig. All diese Striche lassen sich in TeX durch die Eingabe von ein, zwei oder drei aufeinanderfolgenden `-` erzeugen.

```
Ein Bindestrich innerhalb eines
Wortes wie in O-Beine. Ein
Halbgeviertstrich -- auch
Streckenstrich genannt, wird bei
Bereichsangaben wie 4--5 Stunden
oder -- wie dieser -- als
Gedankenstrich zur Abgrenzung von
Teils"atzen benutzt. Der Geviertstrich
--- wird in deutschen Texten im
allgemeinen nicht verwendet, kann
aber in Tabellen bei fehlenden
Angaben sinnvoll sein. Im Englischen
wird er als Gedankenstrich verwendet
und ohne umgebende Leerzeichen
eingegeben.
```

Ein Bindestrich innerhalb eines Wortes wie in O-Beine. Ein Halbgeviertstrich – auch Streckenstrich genannt, wird bei Bereichsangaben wie 4–5 Stunden oder – wie dieser – als Gedankenstrich zur Abgrenzung von Teilsätzen benutzt. Der Geviertstrich — wird in deutschen Texten im allgemeinen nicht verwendet, kann aber in Tabellen bei fehlenden Angaben sinnvoll sein. Im Englischen wird er als Gedankenstrich verwendet und ohne umgebende Leerzeichen eingegeben.

Das Minuszeichen $-$ gehört *nicht* zu diesen Strichen! Es kann nur im mathematischen Modus verwendet werden.

2.6.4 Auslassungspunkte

Auslassungspunkte werden im Textsatz besonders gesetzt. Es gibt daher den Befehl `\dots`, der die passenden Punkte erzeugt.

```
Nicht Orange, Zitrone, ...,
sondern Orange, Zitrone, \dots
```

Nicht Orange, Zitrone, ..., sondern Orange, Zitrone, …

2.6.5 Noch mehr Leerzeichen

Wenn man verhindern möchte, daß LaTeX zwischen zwei Worten einen Zeilenumbruch erzeugt, so kann man dies durch den Befehl `~`, den man an Stelle eines Leerzeichens eingibt, verhindern. Ein typisches Beispiel ist

```
\dots\ finden Sie auf S.~239.
```

… finden Sie auf S. 239.

2.6.6 Kommentare

Falls TeX in Ihrer Eingabedatei auf ein `%` stößt, so wird dieses Zeichen, alle nachfolgenden der entsprechenden Zeile und alle Leerzeichen am Anfang der folgenden Zeile ignoriert. Dadurch kann man Notizen in die Eingabedatei einfügen, die gar nicht oder noch nicht ausgedruckt werden sollen. Außerdem kann man so ein Zeile beenden ohne ein Leerzeichen zu erzeugen.

```
Dies ist ein% lehrreiches
   Beispiel. Man beachte
den % nicht gewollten
fehlenden Abstand zwischen
% Das sieht niemand!
"`ein"' und "`Beispiel"'!
```

Dies ist einBeispiel. Man beachte den fehlenden Abstand zwischen „ein“ und „Beispiel“!

2.7 Das german-Paket

Wie zuvor beschrieben, ist die Eingabe deutscher Umlaute ein wenig umständlich. Hier kann Ihnen das `german`-Paket helfen (siehe Tabelle 5).

Tabelle 5: Einige Befehle des `german`-Pakets

Eingabe	Ausgabe	Eingabe	Ausgabe
"`	„	"'	“
"a	ä	"A	Ä
"e	ë	"E	Ë
"i	ï	"I	Ï
"o	ö	"O	Ö
"u	ü	"U	Ü
"s	ß		

3 Eigene Befehle und Umgebungen

Befehle und Umgebungen spielen eine entscheidene Rolle bei der Eingabe eines Textes. Jede Struktur, die im Dokument (wiederholt) auftaucht, sollte durch einen eigenen Befehl oder eine eigene Umgebung erzeugt werden. LaTeX kennt bereits eine große Zahl von Befehlen und Umgebungen. Trotzdem kann es vorkommen, daß in Ihrem Text eine Struktur vorkommt, auf die LaTeX nicht vorbereitet ist. In solchen Situationen kann man sich helfen, indem man eigenen Befehle und Umgebungen definiert. Im allgemeinen stehen solche Definitionen in der Präambel.

Im Vergleich zu der bisher vorgestellten Texteingabe ist es relativ kompliziert, eigene Befehle und Umgebungen zu definieren. Daher tasten wir uns ganz vorsichtig an dieses Thema heran.

Wir werden in den folgenden Abschnitten noch Gelegenheit haben, genauer auf eigene Befehle einzugehen und können weitere Beispiele vorstellen.

3.1 Definieren von Befehlen

Die einfachsten Befehle sind solche, die nur einen Text ausgeben. Ein Beispiel ist der Befehl `\pounds`, der das Pfundzeichen £ erzeugt. Einen *neuen* Befehl kann man mit dem Befehl `\newcommand` definieren, der zwei Argumente hat. Das erste ist der Name, den der neue Befehl haben soll. Das zweite die *Befehlsdefinition,* in der man angibt, was LaTeX tun soll, sobald es in Ihrer Eingabedateidatei auf den neuen Befehl trifft.

```
\newcommand{\vhom}{Vektorraumhomomorphismus}
Der \vhom\
ist injektiv.
```

Der Vektorraumhomomorphismus ist injektiv.

Der Befehl `\vhom` ist somit eine Abkürzung für das lange und fehlerträchtige Wort *Vektorraumhomomorphismus.*

Etwas komplizierter sind Befehle, die zwingende Argumente haben sollen. Solche Befehle werden verwendet, wenn man sie in Situationen benutzen will, die sich nur wenig voneinander unterscheiden. Die Teile, die unterschiedlich sind, werden dann als Argumente angegeben. Die Befehle aus Tabelle 3 zur Erzeugung von Akzenten gehören zu dieser Art (mit einem zwingenden Argument).

Ein Beispiel für einen neuen Befehl mit Argument wäre einer, der die Telefonnummer eines Mitarbeiters des Fachbereiches ausgibt. Alle Rufnummern von Mitarbeitern fangen mit der 4123 an, der übrige Teil ist variabel. Unser neuer Befehl – wir nennen ihn `\tel` – soll daher *ein* zwingendes Argument haben, nämlich die Nummer des Anschlusses.

```
\newcommand{\tel}[1]{4123-#1}
Herr Nitschke hat die
Telefonnummer~\tel{4929},
Herr Ripken~\tel{5108}.
```

Herr Nitschke hat die Telefonnummer 4123-4929, Herr Ripken 4123-5108.

Hier sieht die Verwendung des `\newcommand`-Befehls etwas anders aus als zuvor. Und zwar tritt ein optionales Argument (das in eckigen Klammern) auf. Dieses gibt an, wieviele Argumente der neue Befehl `\tel` haben soll. Bis zu neun Argumente sind möglich.

In der Befehlsdefinition taucht das Zeichen `#` auf. Es wurde bereits erwähnt, daß es eine besondere Bedeutung für LaTeX hat. Es kennzeichnet einen *Parameter,* der von LaTeX bei Verwendung des Befehls durch das entsprechende Argument des Befehls ersetzt wird: `#1` wird durch das erste Argument ersetzt, `#2` durch das zweite und so weiter.

Die Eingabezeile

```
\newcommand{\tel}[1]{4123-#1}
```

besagt also:

> Definiere einen neuen Befehl `\tel`, der ein Argument hat. Wenn dieser Befehl benutzt wird, so gebe `4123-` aus und anschließend das, was als erstes Argument angegeben wurde.

Zusätzlich zu `\newcommand` gibt es noch den Befehl `\renewcommand`, den man fast genauso verwendet wie `\newcommand`. Der Unterschied besteht darin, daß man mit `\renewcommand` einen bereits vorhandenen Befehl verändern kann.

3.2 Definieren von Umgebungen

Bisher kennen wir nur *eine* Umgebung: `document`. Daher können wir hier nur ein langweiliges Beispiel geben.

Mit dem Befehl

`\newenvironment{`⟨*Name*⟩`}{`⟨*Anfang*⟩`}{`⟨*Ende*⟩`}`

können Sie eine neue Umgebung ⟨*Name*⟩ definieren. Sie haben dadurch die Möglichkeit, in Ihrem Text

`\begin{`⟨*Name*⟩`}` … `\end{`⟨*Name*⟩`}`

einzugeben. Das zweite Argument gibt an, was LaTeX tun soll, sobald es in Ihrer Eingabedatei auf `\begin{`⟨*Name*⟩`}` trifft. Das dritte Argument sagt LaTeX, was bei `\end{`⟨*Name*⟩`}` zu tun ist.

Man kann auch Umgebungen definieren, die Argumente haben. Dazu verwendet man

> `\newenvironment{`⟨*Name*⟩`}[`⟨*Arg*⟩`]{`⟨*Anfang*⟩`}{`⟨*Ende*⟩`}`.

Das optionale Argument ⟨*Arg*⟩ gibt wie beim Befehl `\newcommand` an, wieviele Argumente die neue Umgebung haben soll. Auch hier sind maximal neun Argumente möglich, auf die man durch die Parameter `#1`, … , `#9` zugreifen kann.

Die Parameter `#1`, … , `#9` können *nur* in ⟨*Anfang*⟩ verwendet werden, *nicht* in ⟨*Ende*⟩!

Als Beispiel soll eine neue Umgebung `erkl` definiert werden, die man benutzt, wenn man ein Wort erklärt. Das Wort soll in Gänsefüßchen eingeschlossen werden und ein Doppelpunkt soll sich anschließen.

Die neue Umgebung `erkl` muß somit *ein* Argument haben – nämlich das zu erklärende Wort. Die eigentliche Erklärung ist der Text *zwischen* `\begin{erkl}` und `\end{erkl}`.

Im Argument ⟨*Anfang*⟩ des `\newenvironment`-Befehls legen wir fest, daß das Argument der neuen Umgebung `erkl` zwischen Gänsefüßchen gesetzt wird. Außerdem fügen wir einen Doppelpunkt ein. Am Ende der Umgebung ist nichts zu tun.

```
\newenvironment{erkl}[1]{"`#1"':}{}
\begin{erkl}{Gnu}
  S"ud- und ostafrikanische Antilope.
\end{erkl}
```

„Gnu“: Süd- und ostafrikanische Antilope.

Bestehende Umgebungen lassen sich mit dem Befehl `\renewenvironment` verändern.

4 Strukturen

4.1 Titelinformationen

Am Anfang eines Textes stehen im allgemeinen sein Titel, die Namen der Verfasser(innen) und das Erstellungs- oder Erscheinungsdatum.

LaTeX stellt Ihnen zur Angabe dieser Informationen die Befehle

\title{⟨*Titel*⟩}
\author{⟨*Verfasser*⟩}
\date{⟨*Datum*⟩}

zur Verfügung. Mehrere Autoren werden im Argument des `\author`-Befehls durch `\and` voneinander getrennt.

In der Eingabedatei zu diesem Skript haben wir beispielsweise folgendes geschrieben:

```
\title{Kompaktkurs zu \LaTeXe}
\author{Dirk Nitschke \and Wolfgang Ripken}
```

Wir haben also den `\date`-Befehl weggelassen. In diesem Fall tut LaTeX so, als hätten wir den `\date`-Befehl mit dem Datum des letzten LaTeX-Durchlaufs als Argument aufgerufen.

Keiner der Befehle `\title`, `\author` und `\date` erzeugt irgendwelchen Text. Ausgegeben werden die Titelinformationen erst durch den Befehl `\maketitle`, der nicht vor `\begin{document}` aufgerufen werden darf.

Auf den ersten Blick erscheint dieses Vorgehen vielleicht etwas kompliziert: Wieso soll man bei der Angabe mehrerer Autoren den `\and`-Befehl benutzen und nicht einfach „und" oder „and" oder „," schreiben? Wieso erzeugen die Befehle `\title`, `\author` und `\date` keinen Text? Der Grund besteht im *logischen* Design: Weil die Befehle `\title`, `\author` und `\date` keinen Text erzeugen, können wir sie in beliebiger Reihenfolge aufrufen, ohne daß dies Einfluß auf das Aussehen der Titelseite hätte. Und ob der Befehl `\and` auf der Titelseite das Wörtchen „und" oder „and", ein Komma oder einfach nur einen etwas größeren Abstand erzeugen soll, braucht uns nicht zu kümmern. Dies ist in der jeweiligen Dokumentenklasse festgelegt. (Eigentlich doch ganz praktisch, oder?)

4.2 Der Abstract

Eine weitere meist zu Beginn des Textes auftretende Struktur ist der *Abstract.* Dieser läßt sich mit der Umgebung `abstract` verwirklichen. Der Abstract erhält automatisch eine Überschrift, die durch `\abstractname` erzeugt wird. Das `german`-Paket definiert `\abstractname` als „Zusammenfassung".

```
\begin{abstract}
 In diesem Kurs soll eine
 Einf"uhrung in \LaTeXe\
 gegeben werden.
\end{abstract}
```

Zusammenfassung

In diesem Kurs soll eine Einführung in LaTeX 2_ε gegeben werden.

4.3 Gliederungseinheiten

In einem Text werden Sätze zu Absätzen zusammengefaßt und diese wiederum in eine Abschnittsstruktur eingefügt. Wir kennen zum Beispiel *Teile, Kapitel, Abschnitte* und *Paragraphen.* Wir nennen all diese Einheiten *Gliederungseinheiten.* In diesem Unterabschnitt (sic!) des Skriptes soll beschrieben werden, wie man LaTeX diese Strukturen mitteilt.

Gliederungseinheiten sind meistens mit einer Überschrift versehen, häufig zusätzlich mit einer Nummer. In LaTeX beginnt man jede Gliederungseinheit mit einem Befehl, der als Argument die zugehörige Überschrift hat. Die Nummer wird automatisch erzeugt.

Die Überschrift dieses Unterabschnittes und der erste Satz wurden so erzeugt:

```
\subsection{Gliederungseinheiten}

In einem Text werden S"atze zu Abs"atzen
zusammengefa"st und diese wiederum in eine
Abschnittsstruktur eingef"ugt.
```

Welche Gliederungsbefehle zur Verfügung stehen und wie die Numerierung der Gliederungseinheiten vorzunehmen ist, wird durch die verwendete Dokumentenklasse bestimmt. In den Standardklassen gibt es die folgenden Befehle:

`\chapter`	(nicht in der Klasse `article`)
`\section`	
`\subsection`	
`\subsubsection`	
`\paragraph`	
`\subparagraph`	

Zusätzlich gibt es den Befehl `\part`, mit dem Sie Ihr Dokument in *Teile* unterteilen können.

Nicht immer ist die Numerierung einer Gliederungseinheit gewünscht. Die oben genannten Gliederungsbefehle können auch in der Sternform verwendet werden. Sie erzeugen dann *keine* Nummer.

```
\subsection*{Noch einer}

Diesmal erzeugt \LaTeX\ keine
Nummer.
```

Noch einer

Diesmal erzeugt LaTeX keine Nummer.

Der Sinn der Numerierung von Gliederungseinheiten besteht im allgemeinen darin, an anderer Stelle des Dokumentes auf eine bestimmte Einheit verweisen zu können. Problematisch ist dabei, daß sich im Laufe des Schreibens die Numerierungen häufig ändern und man dann alle Verweise anpassen muß. In Abschnitt 5 (hier haben wir so einen Verweis) erfahren Sie, wie man LaTeX diese Arbeit aufbürden kann.

4.4 Das Inhaltsverzeichnis

Beim Verfassen eines Textes gehört die Erstellung eines Inhaltsverzeichnisses normalerweise zu den aufwendigeren Aufgaben. LaTeX nimmt Ihnen diese Arbeit ab. Wenn Sie ein Inhaltsverzeichnis erzeugen wollen, brauchen Sie lediglich den Befehl `\tableofcontents` an die entsprechende Stelle der Eingabedatei zu schreiben.

In das Inhaltsverzeichnis werden automatisch alle mit den Gliederungsbefehlen aus Unterabschnitt 4.3 erzeugten Überschriften aufgenommen – allerdings nicht diejenigen, die mit den jeweiligen Sternformen angegeben worden sind. Soll eine derartige Überschrift dennoch im Inhaltsverzeichnis erscheinen, müssen Sie dies explizit verlangen, indem Sie direkt *hinter* dem zugehörigen Gliederungsbefehl

`\addcontentsline{toc}{`⟨*Gliederungseinheit*⟩`}{`⟨*Überschrift*⟩`}`

eingeben. Dabei setzen Sie für ⟨*Gliederungseinheit*⟩ die Art der Gliederungseinheit (also beispielsweise `chapter` oder `subsection`) ein.

Das Inhaltsverzeichnis erhält automatisch eine Überschrift, die durch den Befehl `\contentsname` erzeugt wird. Im `german`-Paket wird `\contentsname` als „Inhaltsverzeichnis“ festgelegt.

Um ein korrektes Inhaltsverzeichnis zu erhalten, müssen Sie LaTeX mindestens dreimal durchlaufen lassen!

Das liegt an der Art, wie LaTeX das Inhaltsverzeichnis erzeugt: Wenn LaTeX in der Eingabedatei auf einen Gliederungsbefehl stößt, schreibt es einen Eintrag in eine spezielle Datei (die `toc`-Datei), der im allgemeinen aus der Numerierung, der Überschrift und der aktuellen Seitenzahl besteht. Beim nächsten Durchlauf wird diese Datei durch den Befehl `\tableofcontents` eingelesen und das Inhaltsverzeichnis ausgegeben. Da die `toc`-Datei stets nur die Informationen des vorherigen LaTeX-Laufs enthält, das Inhaltsverzeichnis aber unter Umständen die Anzahl der Seiten vergrößert und damit den nachfolgenden Text verschiebt, müssen Sie LaTeX nun noch ein weiteres Mal durchlaufen lassen.

5 Querverweise

Gliederungseinheiten, Abbildungen, Tabellen und auch mathematische Formeln werden häufig numeriert, damit man im Text auf sie verweisen kann. Dies ist nicht sonderlich kompliziert, wenn man seinen Text nicht mehr verändert und sich die Numerierung nicht mehr ändert. In der Praxis kommt es aber immer wieder vor, daß ein Abschnitt, eine Tabelle oder Formel eingefügt wird. Hierdurch verändert sich die Numerierung nachfolgender Objekte und die Verweise müssen angepaßt werden.

LaTeX bietet die Möglichkeit, die Numerierungen und Verweise automatisch richtig vorzunehmen. Das funktioniert so: Man ordnet einem Objekt, auf das man verweisen möchte, ein sogenanntes *Schlüsselwort* zu. Über dieses Schlüsselwort kann man sich dann im Text auf das Objekt beziehen. LaTeX wandelt diese Referenz in die entsprechende Nummer um.

Die Vergabe der Schlüsselworte wird mit dem Befehl `\label` vorgenommen, die Ausgabe der Nummer durch den Befehl `\ref`. Zusätzlich gibt es noch den Befehl `\pageref`, mit dem Sie auf die Seite verweisen können, auf der sich das Objekt befindet. Ein Schlüsselwort darf aus Groß- und Kleinbuchstaben, Ziffern und Satzzeichen bestehen. Da Groß- und Kleinschreibung von LaTeX unterschieden wird, sind die Schlüsselworte `sec:inhalt` und `sec:Inhalt` verschieden.

Im normalen Text wird dem Schlüsselwort durch den Befehl `\label` die Nummer der aktuellen Gliederungseinheit zugewiesen. In einer numerierten Umgebung die Nummer dieser Umgebung.

```
\section{Querverweise}
\label{sec:refer}
...
Wir befinden uns in
Abschnitt~\ref{sec:refer}, der
auf Seite~\pageref{sec:refer}
begonnen hat.
```

Wir befinden uns in Abschnitt 5, der auf Seite 21 begonnen hat.

6 Literaturangaben

Bei einer Literaturangabe handelt es sich um einen Querverweis auf eine andere Veröffentlichung, eine *Quelle*. Im allgemeinen werden alle in einem Text verwendeten Quellen am Ende des Textes in einem Literaturverzeichnis gesammelt und auf bestimmte Weise numeriert. Zusätzlich versieht man die Quellen mit einem Schlüssel, durch den man auf ihre jeweiligen Nummern zugreifen kann.

6.1 Literaturverzeichnis

Um ein Literaturverzeichnis zu erstellen, wird die `thebibliography`-Umgebung verwendet. Wie diese Umgebung benutzt wird, läßt sich am einfachsten anhand eines Beispiels erläutern:

```
\begin{thebibliography}{2}
\bibitem{higham:1993}
  Nicholas~J. Higham.
  "‘Handbook of Writing for the
  Mathematical Sciences"’.
  SIAM, Philadelphia, 1993.

\bibitem{lamport:1995}
  Leslie Lamport.
  "‘Das \LaTeX-Handbuch"’.
  Addison-Wesley, Bonn, 1995.
\end{thebibliography}
```

Literatur

[1] Nicholas J. Higham. „Handbook of Writing for the Mathematical Sciences“. SIAM, Philadelphia, 1993.

[2] Leslie Lamport. „Das LaTeX-Handbuch“. Addison-Wesley, Bonn, 1995.

Das Argument der `thebibliography`-Umgebung bestimmt, wieviel Platz für die Nummern der einzelnen Literaturangaben gelassen wird. Innerhalb der Umgebung beginnt jeder Eintrag mit einem `\bibitem`-Befehl, der als Argument den Schlüssel der Literaturangabe hat. Die Schlüsselworte dürfen hierbei aus Groß- und Kleinbuchstaben, Ziffern und den Satzzeichen mit Ausnahme des Kommas `,` bestehen.

Das Literaturverzeichnis erhält automatisch eine Überschrift, die in der Klasse `article` durch den Befehl `\refname`, in den Klassen `report` und `book` durch `\bibname` erzeugt wird. Das `german`-Paket definiert `\refname` und `\bibname` als „Literatur“ bzw. „Literaturverzeichnis“.

Das Literaturverzeichnis wird *nicht* automatisch in das Inhaltsverzeichnis übernommen. (Erinnern Sie sich noch an den Befehl `\addcontentsline`?)

Normalerweise werden Literaturangaben mit Zahlen numeriert. Wenn Sie eine andere Markierung (beispielsweise Anfangsbuchstaben der Autoren zusammen mit Erscheinungsjahr) wünschen, können Sie diese als optionales Argument des `\bibitem`-Befehls angeben. Denken Sie auch daran, das Argument der `thebibliography`-Umgebung anzupassen, damit LaTeX ausreichend Platz für die längste Markierung läßt.

```
\begin{thebibliography}{Lam95}
\bibitem[Hig93]{higham:1993}
  Nicholas~J. Higham.
  "‘Handbook of Writing for the
  Mathematical Sciences"’.
  SIAM, Philadelphia, 1993.

\bibitem[Lam95]{lamport:1995}
  Leslie Lamport.
  "‘Das \LaTeX-Handbuch"’.
  Addison-Wesley, Bonn, 1995.
\end{thebibliography}
```

Literatur

[Hig93] Nicholas J. Higham. „Handbook of Writing for the Mathematical Sciences“. SIAM, Philadelphia, 1993.

[Lam95] Leslie Lamport. „Das LaTeX-Handbuch“. Addison-Wesley, Bonn, 1995.

6.2 Referenzen auf Literaturangaben

Für eine Referenz auf eine Literaturangabe gibt es den Befehl `\cite`. Wie der Befehl `\ref` hat `\cite` ein Argument, nämlich den Schlüssel der gewünschten Literaturangabe. `\cite` kann mehr als `\ref`: Es ist möglich, mehrere Quellen gleichzeitig zu zitieren, indem man mehrere Schlüssel durch Kommata voneinander trennt. (Dies ist der Grund, warum das Komma nicht im Schlüsselwort einer Quelle vorkommen darf.) Außerdem hat `\cite` ein optionales Argument, durch das der Querverweis um weitere Informationen wie zum Beispiel Seitenangaben ergänzt werden kann.

```
Die B"ucher~\cite{higham:1993,lamport:1995}
m"ochten wir Ihnen besonders ans Herz legen.
In \cite[S.~81ff.]{lamport:1995}
wird ausf"uhrlich auf die Erstellung des
Literaturverzeichnisses eingegangen.
```

Die Bücher [5, 9] möchten wir Ihnen besonders ans Herz legen. In [9, S. 81ff.] wird ausführlich auf die Erstellung des Literaturverzeichnisses eingegangen.

7 Listen

Listen treten meist in einer der folgenden drei Formen auf:

- Aufzählungen;
- Auflistungen, in denen die Reihenfolge der Einträge nicht wichtig ist (die Einträge sind dann nicht numeriert);
- Beschreibungen.

Für diese Arten von Listen bietet LaTeX die Umgebungen `enumerate`, `itemize` und `description` an. Jeder neue Punkt einer solchen Liste wird durch den Befehl `\item` eingeleitet und automatisch mit einer Standardmarkierung versehen. Man kann solche Listen auch ineinander schachteln, wobei mindestens vier Ebenen zulässig sind. Möchte man für einen Listenpunkt eine andere Markierung verwenden, so gibt man den gewünschten Text als optionales Argument des entsprechenden `\item`-Befehls an.

```
\begin{itemize}
\item Eine Auflistung.
  \begin{enumerate}
  \item Der erste Punkt einer
    Aufz"ahlung.
  \item Eine Liste sollte mindestens
    zwei Punkte enthalten.
  \end{enumerate}
\item[*] Ein Punkt mit anderer
  Markierung.
\end{itemize}
```

- Eine Auflistung.
  1. Der erste Punkt einer Aufzählung.
  2. Eine Liste sollte mindestens zwei Punkte enthalten.

* Ein Punkt mit anderer Markierung.

Die Standardmarkierung der einzelnen Listenpunkte wird – je nach Schachtelungstiefe – bei der `enumerate`-Umgebung durch die Befehle `\labelenumi`, `\labelenumii`, `\labelenumiii` und `\labelenumiv` erzeugt. In der `itemize`-Umgebung geschieht dies durch die Befehle `\labelitemi`, `\labelitemii`, `\labelitemiii` und `\labelitemiv`.

Es lassen sich nur Listenpunkte einer `enumerate`-Umgebung, die mit der Standardmarkierung versehen sind, zufriedenstellend referenzieren.

```
\begin{enumerate}
\item\label{en:def1} Der erste Punkt.
\item[*]\label{en:stern} Ein Stern.
\item\label{en:def2} Der letzte Punkt.
\end{enumerate}
Die Punkte~\ref{en:def1},
\ref{en:stern} (das klappt
leider nicht) und~\ref{en:def2}.
```

1. Der erste Punkt.

* Ein Stern.

2. Der letzte Punkt.

Die Punkte 1, 1 (das klappt leider nicht) und 2.

Die `description`-Umgebung ist für Beschreibungen gedacht. Als Markierung eines Listenpunktes verwendet man meist das Wort, das in diesem Punkt beschrieben wird. Da es daher keine sinnvolle Standardmarkierung geben kann, ist sie leer und man gibt den zu beschreibenden Begriff als optionales Argument des `\item`-Befehls an.

```
\begin{description}
\item[enumerate] Aufz"ahlungen
\item[itemize] Auflistungen
\item[description] Beschreibungen
\item Eine leere Markierung
\end{description}
```

enumerate Aufzählungen

itemize Auflistungen

description Beschreibungen

Eine leere Markierung

8 Theoreme und ähnliches

In mathematischen Texten treten häufig die Strukturen *Satz, Lemma* oder *Beweis* auf. Aber auch in nicht-mathematischen Texten kann es etwas in dieser Art geben: *Regeln, Vermutungen* oder *Schlußfolgerungen.* Diese Strukturen haben etwas gemein: Es handelt sich um *Umgebungen,* denen ein Wort vorangestellt wird, welches angibt, um welche Struktur es sich handelt. Häufig werden diese Strukturen auch numeriert.

Da es unmöglich ist, für alle denkbaren Möglichkeiten eine Umgebung bereitzustellen, gibt es im `amsthm`-Paket, den Befehl `\newtheorem`. Dieser hat zwei Argumente: den Namen der Umgebung und den Text, der als *Kopf* erscheinen soll.

```
\newtheorem{satz}{Satz}
\begin{satz}
  Die Menge der rationalen
  Zahlen ist abz"ahlbar.
\end{satz}
```

Satz 1. *Die Menge der rationalen Zahlen ist abzählbar.*

LaTeX nimmt automatisch eine Numerierung vor und gibt hinter der Nummer einen Punkt aus. Verwendet man die Sternform `\newtheorem*`, so wird die Umgebung *nicht* numeriert.

```
\newtheorem*{cor}{Korollar}
\begin{cor}
  Die Menge der ganzen
  Zahlen ist abz"ahlbar.
\end{cor}
```

Korollar. *Die Menge der ganzen Zahlen ist abzählbar.*

Häufig werden unterschiedliche Arten von solchen theoremähnlichen Strukturen unterschiedlich gesetzt. Bei Theoremen und Sätzen ist es üblich, das Wort *Theorem* bzw. *Satz* fett zu setzen und die Formulierung kursiv. *Definitionen* hingegegen werden aufrecht gesetzt und das Wort *Definition* fett. (Das zu definierende Wort wird im allgemeinen hervorgehoben. Hierzu dient der Befehl `\emph{`⟨*Wort*⟩`}`.) Um deutlich zu machen, zu welcher Sorte von theoremähnlichen Strukturen eine neue Umgebung gehört, gibt es den Befehl `\theoremstyle`. Dieser hat ein Argument, in dem man den entsprechenden Stil angibt. Mögliche Stile sind `plain`, `definition` und `remark`. Wird `\theoremstyle` nicht angegeben, so wird automatisch `plain` gewählt.

```
\theoremstyle{definition}
\newtheorem{defi}{Definition}
\theoremstyle{plain}
\newtheorem{satz}{Satz}
\theoremstyle{remark}
\newtheorem{rem}{Bemerkung}
\begin{satz}
  Das Einselement ist
  eindeutig bestimmt.
\end{satz}
\begin{defi}
  Eine \emph{Gruppe} ist ein
  Monoid, in dem jedes
  Element invertierbar ist.
\end{defi}
\begin{rem}
  Das Einselement ist zu
  sich selbst invers.
\end{rem}
```

Satz 2. *Das Einselement ist eindeutig bestimmt.*

Definition 1. Eine *Gruppe* ist ein Monoid, in dem jedes Element invertierbar ist.

Bemerkung 1. Das Einselement ist zu sich selbst invers.

Im `amsthm`-Paket ist bereits eine Umgebung `proof` definiert, die für *Beweise* verwendet werden kann. Als Kopf wird der durch den Befehl `\proofname` definierte Text ausgegeben. Standardmäßig ist dies „Proof“. Für deutsche Texte müssen Sie diesen Befehl mithilfe von `\renewcommand` ändern. Außerdem wird am Ende einer `proof`-Umgebung ein *Beweisendezeichen* `\qedsymbol` gesetzt. Standardmäßig ist dies „□“.

```
\renewcommand{\proofname}{Beweis}
\begin{proof}
  Der Beweis wird als
  "Ubungsaufgabe gestellt.
\end{proof}
```

Beweis. Der Beweis wird als Übungsaufgabe gestellt. □

Wenn der Beweis zum Beispiel mit einer Liste endet, so wird das Beweisendezeichen an der falschen Stelle gesetzt. In diesem Fall muß man das Zeichen mit dem Befehl `\qedsymbol` oder `\qed` selber an der geeigneten Stelle setzen und anschließend den Befehl `\qed` zum Nichtstun verurteilen.

```
\begin{proof}
  Offensichtlich gilt:
  \begin{itemize}
  \item Eins ist klar!
  \item Zwei auch!\qed
  \end{itemize}
  \renewcommand{\qed}{}
\end{proof}
```

Beweis. Offensichtlich gilt:

- Eins ist klar!
- Zwei auch! □

9 Fußnoten

Im laufenden Text erzeugt man eine Fußnote durch den Befehl `\footnote`. Als Argument gibt man den gewünschten Text der Fußnote an.

```
Man kann auch Verweise auf
Fu"snoten\footnote{\label{fn:diese}%
Wie diese.} verwenden. Das war gerade
Fu"snote~\ref{fn:diese}.
```

Man kann auch Verweise auf Fußnoten[1] verwenden. Das war gerade Fußnote 1.

[1] Wie diese.

Der Befehl `\footnote` funktioniert nicht überall!

10 Mathematik

In diesem Abschnitt möchten wir Ihnen einen ersten Einblick in das Setzen mathematischer Formeln mit LaTeX 2_ε geben. Wir möchten dies anhand des folgenden Beispieltextes tun:

> Wir sagen, eine Folge $(a_n)_{n\in\mathbb{N}}$ reeller Zahlen *konvergiert* gegen eine reelle Zahl a, wenn es zu jedem $\epsilon > 0$ einen Index n_ϵ gibt, so daß
>
> $$|a_n - a| < \epsilon \quad \text{für alle } n > n_\epsilon$$
>
> ist. Wir schreiben dann $a_n \to a$ oder auch $\lim_{n\to\infty} a_n = a$.
>
> Eine Reihe $\sum_{k=1}^{\infty} a_k$ bezeichnen wir als *konvergent*, wenn die Folge $(\sum_{k=1}^{n} a_k)_{n\in\mathbb{N}}$ ihrer Teilsummen konvergiert.

10.1 Eingebettete Formeln

Ein mathematischer Ausdruck, der im laufenden Text erscheint, wird als *eingebettete Formel* bezeichnet. Wir heben einmal alle eingebetteten Formeln in unserem Beispieltext hervor:

> Wir sagen, eine Folge $(a_n)_{n\in\mathbb{N}}$ reeller Zahlen *konvergiert* gegen eine reelle Zahl a, wenn es zu jedem $\epsilon > 0$ einen Index n_ϵ gibt, so daß
>
> $$|a_n - a| < \epsilon \quad \text{für alle } n > n_\epsilon$$
>
> ist. Wir schreiben dann $a_n \to a$ oder auch $\lim_{n\to\infty} a_n = a$.
>
> Eine Reihe $\sum_{k=1}^{\infty} a_k$ bezeichnen wir als *konvergent*, wenn die Folge $(\sum_{k=1}^{n} a_k)_{n\in\mathbb{N}}$ ihrer Teilsummen konvergiert.

Beachten Sie bitte, daß auch der einzelne Buchstabe a in der zweiten Zeile eine eingebettete Formel ist.

LaTeX-Code, der eingebettete Formeln erzeugen soll, wird in `$`-Zeichen eingeschlossen. Das `$`-Zeichen schaltet zwischen dem Text- und dem mathematischen Modus von TeX (dem Programm, auf dem LaTeX basiert) um.

Um ein Gefühl für TeXs mathematischen Modus zu bekommen, schauen wir mal, wie TeX die folgenden Eingaben umsetzt:

`$1+1=2$`	$1 + 1 = 2$
`$1-1=0$`	$1 - 1 = 0$

Das klappt wie erwartet.

`$1 + 1 = 2$`	$1+1=2$

Leerzeichen, die im mathematischen Modus eingegeben werden, wirken sich also nicht auf die Ausgabe aus. Leerzeilen sind nicht erlaubt. (Warum sollte man auch mitten in einer Formel einen neuen Absatz beginnen wollen?)

`$g ist parallel zu h$`	$g ist parallel zu h$

Uups. Was ist hier passiert? Wenn im mathematischen Modus ein Buchstabe auftritt, so wird er von TEX als Variablenname aufgefaßt und in einem kursiven Schrifttyp in der normalen Schriftstärke gesetzt. TEX hat also unsere Eingabe als das Produkt der Variablen g, i, s, t, p, a, r, a, l, l, e, l, z, u und h interpretiert.

`$1+1=2:$ Das ist leicht!`	$1+1=2:$ Das ist leicht!

Der Doppelpunkt ist zu weit nach rechts gerückt. Das liegt daran, daß TEX den Doppelpunkt im mathematischen Modus als Relation und nicht als Satzzeichen auffaßt und dementsprechend etwas mehr Platz um ihn herum freiläßt. Besser ist es also, den Doppelpunkt außerhalb der `$`-Zeichen zu setzen:

`$1+1=2$: Das ist leicht!`	$1+1=2$: Das ist leicht!

Auch `.`, `!` und `?` werden von TEX im mathematischen Modus nicht als Satzzeichen verstanden.

☺ Ein guter Merksatz ist: „Satzzeichen gehören nicht in eine eingebettete Formel.“

10.2 Mathematische Symbole

Zurück zum (Beispiel-) Text: Wenn wir uns die eingebetteten Formeln, die darin vorkommen, noch einmal ansehen, stellen wir fest, daß wir nur von einer einzigen bisher wissen, wie wir sie erzeugen können.

a	`$a$`

Tabelle 6: Große griechische Buchstaben

A	`\mathrm{A}`	N	`\mathrm{N}`
B	`\mathrm{B}`	Ξ	`\Xi`
Γ	`\Gamma`	O	`\mathrm{O}`
Δ	`\Delta`	Π	`\Pi`
E	`\mathrm{E}`	P	`\mathrm{P}`
Z	`\mathrm{Z}`	Σ	`\Sigma`
H	`\mathrm{H}`	T	`\mathrm{T}`
Θ	`\Theta`	Υ, Y	`\Upsilon`, `\mathrm{Y}`
I	`\mathrm{I}`	Φ	`\Phi`
K	`\mathrm{K}`	X	`\mathrm{X}`
Λ	`\Lambda`	Ψ	`\Psi`
M	`\mathrm{M}`	Ω	`\Omega`

Diese Befehle funktionieren nur im mathematischen Modus.

Tabelle 7: Kleine griechische Buchstaben

α	`\alpha`	ν	`\nu`
β	`\beta`	ξ	`\xi`
γ	`\gamma`	o	`o`
δ	`\delta`	π, ϖ	`\pi`, `\varpi`
ϵ, ε	`\epsilon`, `\varepsilon`	ρ, ϱ	`\rho`, `\varrho`
ζ	`\zeta`	σ, ς	`\sigma`, `\varsigma`
η	`\eta`	τ	`\tau`
θ, ϑ	`\theta`, `\vartheta`	υ	`\upsilon`
ι	`\iota`	ϕ, φ	`\phi`, `\varphi`
κ	`\kappa`	χ	`\chi`
λ	`\lambda`	ψ	`\psi`
μ	`\mu`	ω	`\omega`

Diese Befehle funktionieren nur im mathematischen Modus.

Alle übrigen Formeln enthalten Symbole, die auf unserer Tastatur nicht vorhanden sind.

Aber keine Panik: Zum Erzeugen von Symbolen, für die es keine Taste gibt, gibt es in TeX und LaTeX Befehle.

Zum Beispiel existieren Befehle zum Setzen griechischer Buchstaben. Diese stellen wir in den Tabellen 6 und 7 zusammen.

Mit Hilfe des Befehls `\epsilon` aus Tabelle 7 können wir nun schon eine weitere Formel aus unserem Beispieltext erzeugen:

$\epsilon > 0$	`$\epsilon>0$`

Alle übrigen mathematischen Symbole, die in LaTeX 2_ε definiert sind, listen wir zusammen mit den Befehlen, durch die Sie diese erzeugen können, in Tabelle 8 auf (was es mit der dritten Spalte dieser Tabelle auf sich hat, erklären wir in Abschnitt 10.8).

Tabelle 8: Mathematische Symbole

<table>
<tr><th>Symbol</th><th>Befehl</th><th>Klasse</th></tr>
<tr><td>$.$</td><td>.</td><td>Ord</td></tr>
<tr><td>$\ldotp$</td><td>\ldotp</td><td>Punct</td></tr>
<tr><td>$\cdot$</td><td>\cdot</td><td>Bin</td></tr>
<tr><td>$\cdotp$</td><td>\cdotp</td><td>Punct</td></tr>
<tr><td>$/$</td><td>/</td><td>Ord</td></tr>
<tr><td>$\backslash$</td><td>\backslash</td><td>Ord</td></tr>
<tr><td>$\setminus$</td><td>\setminus</td><td>Bin</td></tr>
<tr><td>$|$</td><td>| oder \vert</td><td>Ord</td></tr>
<tr><td>$\mid$</td><td>\mid</td><td>Rel</td></tr>
<tr><td>$\|$</td><td>\| oder \Vert</td><td>Ord</td></tr>
<tr><td>$\parallel$</td><td>\parallel</td><td>Rel</td></tr>
<tr><td>$\emptyset$</td><td>\emptyset</td><td>Ord</td></tr>
<tr><td>$\aleph$</td><td>\aleph</td><td>Ord</td></tr>
<tr><td>$\imath$</td><td>\imath</td><td>Ord</td></tr>
<tr><td>$\jmath$</td><td>\jmath</td><td>Ord</td></tr>
<tr><td>ℓ</td><td>\ell</td><td>Ord</td></tr>
<tr><td>$\wp$</td><td>\wp</td><td>Ord</td></tr>
<tr><td>$\Re$</td><td>\Re</td><td>Ord</td></tr>
<tr><td>$\Im$</td><td>\Im</td><td>Ord</td></tr>
<tr><td>∂</td><td>\partial</td><td>Ord</td></tr>
<tr><td>∞</td><td>\infty</td><td>Ord</td></tr>
<tr><td>$\prime$</td><td>\prime</td><td>Ord</td></tr>
<tr><td>$\top$</td><td>\top</td><td>Ord</td></tr>
<tr><td>$\bot$</td><td>\bot</td><td>Ord</td></tr>
</table>

Mathematische Symbole (Forts.)

Symbol	Befehl	Klasse
$\perp$	`\perp`	Rel
$\angle$	`\angle`	Ord
$\triangle$	`\triangle`	Ord
∇	`\nabla`	Ord
$\forall$	`\forall`	Ord
$\exists$	`\exists`	Ord
$\neg$	`\neg`	Ord
$\flat$	`\flat`	Ord
$\natural$	`\natural`	Ord
$\sharp$	`\sharp`	Ord
$\clubsuit$	`\clubsuit`	Ord
$\diamondsuit$	`\diamondsuit`	Ord
$\heartsuit$	`\heartsuit`	Ord
$\spadesuit$	`\spadesuit`	Ord
¶	`\mathparagraph`	Ord
§	`\mathsection`	Ord
$	`\mathdollar`	Ord
$\sum$	`\sum`	Op
$\int$	`\int`	Op
$\oint$	`\oint`	Op
$\prod$	`\prod`	Op
$\amalg$	`\amalg`	Bin
$\coprod$	`\coprod`	Op
$\vee$	`\vee`	Bin
$\bigvee$	`\bigvee`	Op
$\wedge$	`\wedge`	Bin
$\bigwedge$	`\bigwedge`	Op
$\cap$	`\cap`	Bin
$\bigcap$	`\bigcap`	Op
$\cup$	`\cup`	Bin
$\bigcup$	`\bigcup`	Op
$\uplus$	`\uplus`	Bin
$\biguplus$	`\biguplus`	Op
$\sqcap$	`\sqcap`	Bin
$\sqcup$	`\sqcup`	Bin
$\bigsqcup$	`\bigsqcup`	Op
$\otimes$	`\otimes`	Bin
$\bigotimes$	`\bigotimes`	Op
$\oplus$	`\oplus`	Bin
$\bigoplus$	`\bigoplus`	Op
$\odot$	`\odot`	Bin

Mathematische Symbole (Forts.)

Symbol	Befehl	Klasse
$\bigodot$	`\bigodot`	Op
$\oslash$	`\oslash`	Bin
$\ominus$	`\ominus`	Bin
$\surd$	`\surd`	Op
$+$	`+`	Bin
$-$	`-`	Bin
$\pm$	`\pm`	Bin
$\mp$	`\mp`	Bin
$\div$	`\div`	Bin
$\times$	`\times`	Bin
$\ast$	`*` oder `\ast`	Bin
$\star$	`\star`	Bin
$\wr$	`\wr`	Bin
$\triangleleft$	`\triangleleft`	Bin
$\triangleright$	`\triangleright`	Bin
$\bigtriangleup$	`\bigtriangleup`	Bin
$\bigtriangledown$	`\bigtriangledown`	Bin
$\ddagger$	`\ddagger`	Bin
$\dagger$	`\dagger`	Bin
$\diamond$	`\diamond`	Bin
$\bullet$	`\bullet`	Bin
$\circ$	`\circ`	Bin
$\bigcirc$	`\bigcirc`	Bin
$<$	`<`	Rel
$>$	`>`	Rel
$\leq$	`\leq` oder `\le`	Rel
$\geq$	`\geq` oder `\ge`	Rel
$\gg$	`\gg`	Rel
$\ll$	`\ll`	Rel
$=$	`=`	Rel
$\neq$	`\neq` oder `\ne`	Rel
$\equiv$	`\equiv`	Rel
$\doteq$	`\doteq`	Rel
$\approx$	`\approx`	Rel
$\simeq$	`\simeq`	Rel
$\cong$	`\cong`	Rel
$\sim$	`\sim`	Rel
$\asymp$	`\asymp`	Rel
$:$	`:`	Rel
$\colon$	`\colon`	Punct
$\in$	`\in`	Rel

Mathematische Symbole (Forts.)

Symbol	Befehl	Klasse
$\ni$	`\ni`	Rel
$\notin$	`\notin`	Rel
$\supset$	`\supset`	Rel
$\subset$	`\subset`	Rel
$\supseteq$	`\supseteq`	Rel
$\subseteq$	`\subseteq`	Rel
$\sqsubseteq$	`\sqsubseteq`	Rel
$\sqsupseteq$	`\sqsupseteq`	Rel
$\propto$	`\propto`	Rel
$\dashv$	`\dashv`	Rel
$\vdash$	`\vdash`	Rel
$\models$	`\models`	Rel
$\bowtie$	`\bowtie`	Rel
$\smile$	`\smile`	Rel
$\frown$	`\frown`	Rel
$\succ$	`\succ`	Rel
$\prec$	`\prec`	Rel
$\succeq$	`\succeq`	Rel
$\preceq$	`\preceq`	Rel
$\leftarrow$	`\leftarrow` oder `\gets`	Rel
$\longleftarrow$	`\longleftarrow`	Rel
$\hookleftarrow$	`\hookleftarrow`	Rel
$\leftrightarrow$	`\leftrightarrow`	Rel
$\longleftrightarrow$	`\longleftrightarrow`	Rel
$\rightarrow$	`\rightarrow` oder `\to`	Rel
$\longrightarrow$	`\longrightarrow`	Rel
$\hookrightarrow$	`\hookrightarrow`	Rel
$\mapsto$	`\mapsto`	Rel
$\longmapsto$	`\longmapsto`	Rel
$\Leftarrow$	`\Leftarrow`	Rel
$\Longleftarrow$	`\Longleftarrow`	Rel
$\Leftrightarrow$	`\Leftrightarrow`	Rel
$\Longleftrightarrow$	`\Longleftrightarrow`	Rel
$\Rightarrow$	`\Rightarrow`	Rel
$\Longrightarrow$	`\Longrightarrow`	Rel
$\uparrow$	`\uparrow`	Rel
$\updownarrow$	`\updownarrow`	Rel
$\downarrow$	`\downarrow`	Rel
$\Uparrow$	`\Uparrow`	Rel
$\Updownarrow$	`\Updownarrow`	Rel
$\Downarrow$	`\Downarrow`	Rel

Mathematische Symbole (Forts.)

Symbol	**Befehl**	**Klasse**
$\nearrow$	`\nearrow`	Rel
$\searrow$	`\searrow`	Rel
$\nwarrow$	`\nwarrow`	Rel
$\swarrow$	`\swarrow`	Rel
$\leftharpoonup$	`\leftharpoonup`	Rel
$\leftharpoondown$	`\leftharpoondown`	Rel
$\rightleftharpoons$	`\rightleftharpoons`	Rel
$\rightharpoonup$	`\rightharpoonup`	Rel
$\rightharpoondown$	`\rightharpoondown`	Rel
$($	`(`	Open
$)$	`)`	Close
$\langle$	`\langle`	Open
$\rangle$	`\rangle`	Close
$\{$	`\{` oder `\lbrace`	Open
$\}$	`\}` oder `\rbrace`	Close
$[$	`[` oder `\lbrack`	Open
$]$	`]` oder `\rbrack`	Close
$\lceil$	`\lceil`	Open
$\rceil$	`\rceil`	Close
$\lfloor$	`\lfloor`	Open
$\rfloor$	`\rfloor`	Close
$!$	`!`	Close
$?$	`?`	Close
$,$	`,`	Punct
$;$	`;`	Punct

Diese Befehle funktionieren nur im mathematischen Modus.

Einige Pakete (wie z. B. `amssymb` oder `latexsym`) stellen Ihnen zusätzlich zu den in Tabelle 8 aufgeführten weitere mathematische Symbole zur Verfügung. Wenn Sie also in Tabelle 8 ein gewisses Symbol vermissen, könnte es in einem dieser Pakete definiert sein.

10.3 Hoch- und Tiefstellungen

In den Formeln unseres Beispieltextes tauchen Ausdrücke auf, die in einem kleineren Schrifttyp etwas höher oder tiefer gesetzt werden. Beispiele sind die Summationsgrenzen $k = 1$ und n sowie der Index k im Ausdruck $\sum_{k=1}^{n} a_k$.

Derartige Hoch- und Tiefstellungen werden in der Eingabedatei in der Form

⟨*Symbol*⟩`^{`⟨*Hochstellung*⟩`}`

bzw.

⟨*Symbol*⟩`_{`⟨*Tiefstellung*⟩`}`

angegeben.

Dies funktioniert nur im mathematischen Modus.

Wir sind nun in der Lage, weitere Formeln aus unserem Beispieltext zu erzeugen:

n_{ϵ}	`$n_{\epsilon}$`
$a_{n}\rightarrow\infty$	`$a_{n}\rightarrow\infty$`
$\sum_{k=1}^{\infty} a_{k}$	`$\sum_{k=1}^{\infty} a_{k}$`

10.4 Lange Funktionen- und Operatornamen

Versuchen wir, noch eine Formel unseres Beispieltextes zu setzen:

`$lim_{n\rightarrow\infty} a_{n}$`	$lim_{n\rightarrow\infty} a_{n}$

Moment, das ging schief (vergleichen Sie das Ergebnis mit den Ausdruck $\lim_{n\rightarrow\infty} a_n$ aus unserem Beispieltext).

Es ist üblich, Funktionen- und Operatornamen, die aus mehreren Buchstaben bestehen, in einem aufrechten Schrifttyp zu setzen. Dadurch sollen Verwechslungen mit Produkten mehrerer Variabler vermieden werden.

In LaTeX 2_{ε} sind eine Reihe langer Funktionen- und Operatornamen vordefiniert (siehe Tabelle 9). Wie Sie weitere Funktionen- und Operatornamen definieren können, erfahren Sie in Unterabschnitt 11.11.

Tabelle 9: Lange Funktionen- und Operatornamen

`\arccos`	`\cos`	`\csc`	`\exp`	`\ker`	`\limsup`	`\min`	`\sinh`
`\arcsin`	`\cosh`	`\deg`	`\gcd`	`\lg`	`\ln`	`\Pr`	`\sup`
`\arctan`	`\cot`	`\det`	`\hom`	`\lim`	`\log`	`\sec`	`\tan`
`\marg`	`\coth`	`\dim`	`\inf`	`\liminf`	`\max`	`\sin`	`\tanh`

Diese Befehle funktionieren nur im mathematischen Modus.

Indem wir nun den Befehl `\lim` aus Tabelle 9 benutzen, erhalten wir:

`$\lim_{n\rightarrow\infty} a_{n}$`	$\lim_{n\rightarrow\infty} a_{n}$

10.5 Variieren der Schriftart von Buchstaben

Wie können wir das Symbol $\mathbb{N}$ für die Menge der natürlichen Zahlen erzeugen? In LaTeX 2$_\varepsilon$ gibt es eine einfache Möglichkeit: Wir verändern die Schriftart des Buchstabens `N`.

Der Befehl

`\mathbb{`⟨*Großbuchstabe*⟩`}`

aus dem **`amsfonts`**-Paket setzt ⟨*Großbuchstabe*⟩ in einem „Blackboard Bold“ Font. Taucht dabei ⟨*Großbuchstabe*⟩ in einem hoch- oder tiefgestellten Formelteil auf, so wird die Schriftgröße automatisch angepaßt.

Dieser Befehl funktioniert nur im mathematischen Modus.

Die beiden verbleibenden eingebetteten Formeln in unserem Beispieltext können wir also so erzeugen:

$(a_n)_{n\in\mathbb{N}}$	`$(a_{n})_{n\in\mathbb{N}}$`
$(\sum_{k=1}^{n} a_k)_{n\in\mathbb{N}}$	`$(\sum_{k=1}^{n} a_{k})_{n\in\mathbb{N}}$`

Es gibt noch eine ganze Reihe weiterer Befehle zum Verändern der Schriftart von Buchstaben im mathematischen Modus (siehe beispielsweise L. Lamport [9, S. 61f. oder S. 221]).

10.6 Abgesetzte Formeln

Formeln, die in einer eigenen Zeile stehen, werden als *abgesetzte Formeln* bezeichnet. In unserem Beispieltext gibt es nur eine abgesetzte Formel. Wir heben sie durch einen grauen Hintergrund hervor:

> Wir sagen, eine Folge $(a_n)_{n\in\mathbb{N}}$ reeller Zahlen *konvergiert* gegen eine reelle Zahl a, wenn es zu jedem $\epsilon > 0$ einen Index n_ϵ gibt, so daß
>
> $$|a_n - a| < \epsilon \quad \text{für alle } n > n_\epsilon$$
>
> ist. Wir schreiben dann $a_n \to a$ oder auch $\lim_{n\to\infty} a_n = a$.
>
> Eine Reihe $\sum_{k=1}^{\infty} a_k$ bezeichnen wir als *konvergent*, wenn die Folge $(\sum_{k=1}^{n} a_k)_{n\in\mathbb{N}}$ ihrer Teilsummen konvergiert.

Das `amsmath`-Paket stellt Ihnen die Umgebungen `equation` und `equation*` zum Setzen abgesetzter Formeln zur Verfügung. Die `*`-Form dieser Umgebung erzeugt im Gegensatz zur Form ohne `*` keine Formelnummer. Einfache Beispiele:

```
\begin{equation*}
  a^{2}+b^{2}=c^{2}
\end{equation*}
```

$$a^2 + b^2 = c^2$$

```
\begin{equation}
  a^{2}+b^{2}=c^{2}
\end{equation}
```

$$a^2 + b^2 = c^2 \qquad (1)$$

10.7 Text in Formeln

Die abgesetzte Formel enthält neben mathematischen Symbolen auch Text. Das `amsmath`-Paket stellt Ihnen den Befehl

`\text{`⟨*Text*⟩`}`

zum Setzen von Text in mathematischen Formeln zur Verfügung. Beispiel:

```
\begin{equation*}
  f(x)>0
  \quad
  \text{fast "uberall}
\end{equation*}
```

$$f(x) > 0 \quad \text{fast überall}$$

In diesem Beispiel ist der Befehl `\quad` für den Zwischenraum zwischen „0“ und „fast“ verantwortlich. Ohne diesen Befehl erhielte man:

$$f(x) > 0\text{fast überall}$$

Die Schriftgröße des Textes, den der Befehl `\text{`⟨*Text*⟩`}` erzeugt, wird automatisch angepaßt:

```
\begin{equation*}
  \sum_{\text{$n$ ungerade}}2^{-n}
\end{equation*}
```

$$\sum_{\text{n ungerade}} 2^{-n}$$

10.8 Klassen mathematischer Symbole

Jetzt ist es an der Zeit, Ihnen endlich zu erzählen, welche Bewandtnis es mit der dritten Spalte von Tabelle 8 hat. TeX teilt alle mathematischen Symbole in die folgenden Klassen ein:

Ord gewöhnliche Symbole;

Op große Operatorsymbole wie $\sum$;

Bin zweistellige Operatoren wie $+$ oder $-$;

Rel Relationen wie $<$, $>$ oder $=$;

Open öffnende Klammern;

Close schließende Klammern;

Punct Trennzeichen wie zum Beispiel die Kommas in (a, b, c).

Aufgrund der Zugehörigkeit eines Symbols zu einer dieser Klassen entscheidet TeX beispielsweise über den Abstand dieses Symbols zu den umgebenden Formelteilen, oder ob eine eingebettete Formel an einer bestimmten Stelle umgebrochen werden darf.

Wenn TeX in einer mathematischen Formel ausnahmsweise einmal völlig falsche Abstände erzeugt hat, sollten Sie überprüfen, ob die Befehle, die Sie verwendet haben, Symbole aus den richtigen Klassen erzeugen. Ist dies nicht der Fall, können Sie entweder in Tabelle 8 nachsehen, ob es nicht vielleicht einen Befehl gibt, der das gleiche Symbol in der richtigen Klasse liefert; oder Sie teilen TeX explizit mit, zu welcher Klasse das Symbol, das Sie setzen möchten, gehören soll. Hierzu dienen die Befehle

`\mathord{`⟨*Symbol*⟩`}` (um ⟨*Symbol*⟩ als gewöhnliches Symbol zu setzen),

`\mathop{`⟨*Symbol*⟩`}` (um ⟨*Symbol*⟩ als großes Operatorsymbol zu setzen),

`\mathbin{`⟨*Symbol*⟩`}` (um ⟨*Symbol*⟩ als zweistelligen Operator zu setzen),

`\mathrel{`⟨*Symbol*⟩`}` (um ⟨*Symbol*⟩ als Relation zu setzen),

`\mathopen{`⟨*Symbol*⟩`}` (um ⟨*Symbol*⟩ als öffnende Klammer zu setzen),

`\mathclose{`⟨*Symbol*⟩`}` (um ⟨*Symbol*⟩ als schließende Klammer zu setzen),

`\mathpunct{`⟨*Symbol*⟩`}` (um ⟨*Symbol*⟩ als Trennzeichen zu setzen).

Ein Beispiel: In der abgesetzten Formel in unserem Beispieltext kommt der Ausdruck

$$|a_n - a| < \epsilon$$

vor. Nehmen wir an, wir würden zum Setzen der Betragsstriche den Befehl `\mid` verwenden. Dann würden wir

$$\mid a_n - a \mid < \epsilon$$

erhalten (nicht schön, oder?). Der Befehl `\mid` dient zum Setzen des senkrechten Strichs in Formeln des Typs $a \mid b$ („a teilt b"). Dieser senkrechte Strich ist eine Relation, gehört also zur Klasse Rel. Für Betragsstriche könnten wir `|` direkt eingeben (das geht in diesem Fall gut, obwohl der durch `|` erzeugte Strich zur Klasse Ord gehört). Besser ist es allerdings, den linken Betragsstrich durch `\mathopen{|}` und den rechten durch `\mathclose{|}` zu setzen. Zum Setzen der Formel $\mathopen{|}-x\mathclose{|}=\mathopen{|}+x\mathclose{|}$ *müssen* wir `\mathopen{|}` und `\mathclose{|}` verwenden: Vergleichen Sie

```
$\mathopen{|}-x\mathclose{|}
  =\mathopen{|}+x\mathclose{|}$
```

$\mathopen{|}-x\mathclose{|}=\mathopen{|}+x\mathclose{|}$

und

```
$|-x|=|+x|$
```

$|-x|=|+x|$

Zum guten Schluß: Die abgesetzte Formel in unserem Beispieltext können wir folgendermaßen erzeugen:

$$\mathopen{|} a_{n}-a\mathclose{|} <\epsilon \quad \text{für alle } n>n_{\epsilon}$$

```
\begin{equation*}
  \mathopen{|} a_{n}-a\mathclose{|}
    <\epsilon
  \quad
  \text{f"ur alle $n>n_{\epsilon}$}
\end{equation*}
```

11 Noch mehr Mathematik

11.1 Brüche

Brüche kann man auf zwei Arten schreiben: mit einem schrägen Bruchstrich wie in 2/3 oder einem waagerechten Bruchstrich wie in $\frac{2}{3}$. In eingebetteten Formeln und in Hoch- und Tiefstellungen ist der schräge Bruchstrich zu bevorzugen.

Den schrägen Bruchstrich erzeugt man in LaTeX einfach durch das Zeichen `/`. Beispiel:

```
$2/3$
```

$2/3$

Obwohl ein schräger Bruchstrich aus mathematischer Sicht eigentlich zur Klasse Bin gehört, `/` aber einen Schrägstrich der Klasse Ord liefert, sollten Sie nicht `\mathbin{/}` schreiben. Andernfalls wird zuviel Leerraum um den Bruchstrich gesetzt (vgl. D. Knuth [7, S. 132]).

Einen Bruch mit einem waagerechten Bruchstrich erhält man mit dem Befehl `\frac{`⟨*Zähler*⟩`}{`⟨*Nenner*⟩`}`. Beispiel:

```
\begin{equation*}
  \frac{2}{3}
\end{equation*}
```

$$\frac{2}{3}$$

11.2 Klammersymbole in verschiedenen Größen

Ausdrücke wie

$$||x| - |y|| \leq |x - y|$$

oder

$$-((x(x + 1) - (x + 1))(x + 1)),$$

die aus mehreren ineinander geschachtelten Formelteilen bestehen, sind oft schwer zu verstehen, da es nicht leicht ist, öffnende und schließende Klammern einander zuzuordnen. Hier ist es sinnvoll, Klammern unterschiedlicher Größe zu verwenden: Die Formeln

$$\bigl||x| - |y|\bigr| \leq |x - y|$$

und

$$-\Bigl(\bigl(x(x+1)-(x+1)\bigr)(x+1)\Bigr)$$

sehen schon sehr viel einfacher aus.

Um Klammersymbole zu erhalten, die größer als die standardmäßig erzeugten sind, stellt man öffnenden Klammern einfach einen der Befehle `\bigl`, `\Bigl`, `\biggl` oder `\Biggl` und schließenden Klammern `\bigr`, `\Bigr`, `\biggr` oder `\Biggr` voran.

```
\begin{equation*}
  \Biggl( \biggl( \Bigl( \bigl( (
  ) \bigr) \Bigr) \biggr) \Biggr)
\end{equation*}
```

$$\Biggl(\biggl(\Bigl(\bigl(( ) \bigr) \Bigr) \biggr) \Biggr)$$

Diese Befehle sorgen dann auch gleich dafür, daß die Klammersymbole, auf die sie wirken, zu den richtigen Klassen gehören (`\bigr|` bedeutet also dasselbe wie `\mathclose{\bigr|}`).

Es kann noch einen anderen Grund dafür geben, daß man größere Klammern erzeugen möchte: Die normalerweise erzeugten Klammern sind zu klein, um den Formelteil, der eingeklammert werden soll, zu umschließen. Klammern, die (mindestens) so groß sind wie der Formelteil, den sie umschließen, kann man erzeugen, indem man der öffnenden Klammer ein `\left` und der schließenden Klammer ein `\right` voranstellt.

```
\begin{equation*}
  2\left(
      \frac{
        \sum_{k=0}^{\infty}A^k
      }{
        \sum_{k=0}^{\infty}B^k}
      +1
   \right)
\end{equation*}
```

$$2\left(\frac{\sum_{k=0}^{\infty}A^k}{\sum_{k=0}^{\infty}B^k}+1\right)$$

Die Klammersymbole, auf die Sie die in diesem Unterabschnitt vorgestellten Befehle anwenden können, sind in Tabelle 10 zusammengestellt.

Der schräge Bruchstrich wirkt mitunter etwas dürftig, wenn er neben einer großen Klammer erscheint. Größere Bruchstriche erhalten Sie durch die Befehle `\big/`, `\Big/`, `\bigg/` und `\Bigg/`.

11.3 Mathematische Akzente

Mathematische Akzente (wie beispielsweise Tilden oder Dächer) können Sie durch die Befehle in Tabelle 11 erzeugen. Als Beispiel ist dort der Buchstabe a gewählt worden. Die Befehle funktionieren aber genauso mit jedem anderen Symbol.

Tabelle 10: Klammersymbole

<table>
<tr><td><code>(</code></td><td>$($</td><td><code>\lceil</code></td><td>$\lceil$</td></tr>
<tr><td><code>)</code></td><td>$)$</td><td><code>\rceil</code></td><td>$\rceil$</td></tr>
<tr><td><code>\{</code> oder <code>\lbrace</code></td><td>$\{$</td><td><code>\lfloor</code></td><td>$\lfloor$</td></tr>
<tr><td><code>\}</code> oder <code>\rbrace</code></td><td>$\}$</td><td><code>\rfloor</code></td><td>$\rfloor$</td></tr>
<tr><td><code>\langle</code></td><td>$\langle$</td><td><code>\uparrow</code></td><td>$\uparrow$</td></tr>
<tr><td><code>\rangle</code></td><td>$\rangle$</td><td><code>\updownarrow</code></td><td>$\updownarrow$</td></tr>
<tr><td><code>|</code> oder <code>\vert</code></td><td>$\vert$</td><td><code>\downarrow</code></td><td>$\downarrow$</td></tr>
<tr><td><code>\|</code> oder <code>\Vert</code></td><td>$\Vert$</td><td><code>\Uparrow</code></td><td>$\Uparrow$</td></tr>
<tr><td><code>[</code> oder <code>\lbrack</code></td><td>$[$</td><td><code>\Updownarrow</code></td><td>$\Updownarrow$</td></tr>
<tr><td><code>]</code> oder <code>\rbrack</code></td><td>$]$</td><td><code>\Downarrow</code></td><td>$\Downarrow$</td></tr>
</table>

Diese Befehle funktionieren nur im mathematischen Modus.

Tabelle 11: Mathematische Akzente

$\widetilde{a}$	`\widetilde{a}`	$\check{a}$	`\check{a}`	$\ddot{a}$	`\ddot{a}`
$\tilde{a}$	`\tilde{a}`	$\acute{a}$	`\acute{a}`	$\breve{a}$	`\breve{a}`
$\widehat{a}$	`\widehat{a}`	$\grave{a}$	`\grave{a}`	$\bar{a}$	`\bar{a}`
$\hat{a}$	`\hat{a}`	$\dot{a}$	`\dot{a}`	$\vec{a}$	`\vec{a}`

Diese Befehle funktionieren nur im mathematischen Modus.

Wenn Sie einem i oder j einen Akzent aufsetzen wollen, stört der Punkt. Die Befehle `\imath` und `\jmath` erzeugen ein (punktloses) $\imath$ bzw. $\jmath$.

Sofern ihr Argument höchstens etwa drei Zeichen lang ist, liefern die Befehle `\widetilde{`⟨*Argument*⟩`}` und `\widehat{`⟨*Argument*⟩`}` eine Tilde bzw. ein Dach, das sich über die gesamte Breite von ⟨*Argument*⟩ erstreckt (beispielsweise ergibt `$\widetilde{abc}$` den Ausdruck $\widetilde{abc}$).

11.4 Formelsysteme

Zum Setzen von Formelsystemen, bei denen die einzelnen Formeln in mehreren Zeilen untereinander stehen, stellt Ihnen das `amsmath`-Paket u. a. die Umgebungen `gather`, `align` und `alignat` sowie `gather*`, `align*` und `alignat*` zur Verfügung. Die `*`-Formen dieser Umgebungen erzeugen im Gegensatz zu den Formen ohne `*` keine Formelnummern.

Im folgenden möchten wir diese Umgebungen anhand einfacher Beispiele vorstellen.

11.4.1 Formelsysteme ohne Ausrichtung

Die Umgebungen `gather` und `gather*` erzeugen Formelsysteme, bei denen die einzelnen Formeln zentriert in mehrereren Zeilen untereinander stehen. Zum Trennen der Zeilen dient der Befehl `\\`.

```
\begin{gather}
  a^{2}+b^{2}=c^{2}\\
  a<1
\end{gather}
```

$$a^2 + b^2 = c^2 \qquad (2)$$
$$a < 1 \qquad (3)$$

```
\begin{gather*}
  a^{2}+b^{2}=c^{2}\\
  a<1
\end{gather*}
```

$$a^2 + b^2 = c^2$$
$$a < 1$$

11.4.2 Formelsysteme mit Ausrichtung an einer Position

Die Umgebungen `align` und `align*` erzeugen Formelsysteme, bei denen die einzelnen Formeln in mehreren Zeilen untereinander stehen und zueinander an einer Position ausgerichtet werden. Wie bei der `gather`-Umgebung werden die Zeilen durch den Befehl `\\` getrennt. Die Position, an der die Zeilen ausgerichtet werden sollen, wird durch den Tabulator `&` markiert.

```
\begin{align}
  a^{2}+b^{2}&=c^{2}\\
  a&<1
\end{align}
```

$$\begin{aligned} a^2 + b^2 &= c^2 && (4) \\ a &< 1 && (5) \end{aligned}$$

```
\begin{align*}
  a^{2}+b^{2}&=c^{2}\\
  a&<1
\end{align*}
```

$$
\begin{aligned}
a^2 + b^2 &= c^2 \\
a &< 1
\end{aligned}
$$

11.4.3 Formelsysteme mit Ausrichtung an mehreren Positionen

Die Umgebungen `alignat` und `alignat*` erzeugen Formelsysteme, bei denen die einzelnen Formeln in mehreren Zeilen untereinander stehen und an mehreren Positionen zueinander ausgerichtet werden. Dies ist vor allem für Gleichungssysteme interessant.

Die Benutzung von `alignat` und `alignat*` ist relativ kompliziert und läßt sich wohl am einfachsten anhand eines Beispiels erklären:

```
\begin{alignat}{3}
    a&={}  &  3216b&+{}  &     13c& \\
  17a&={}  &      b&-{}  &  2353c&
\end{alignat}
```

$$
\begin{alignedat}{3}
a &={} & 3216b &+{} & 13c & \qquad (6) \\
17a &={} & b &-{} & 2353c & \qquad (7)
\end{alignedat}
$$

```
\begin{alignat*}{3}
    a&={}  &  3216b&+{}  &     13c& \\
  17a&={}  &      b&-{}  &  2353c&
\end{alignat*}
```

$$
\begin{alignedat}{3}
a &={} & 3216b &+{} & 13c & \\
17a &={} & b &-{} & 2353c &
\end{alignedat}
$$

Hier wird das Gleichungssystem durch den zweiten und vierten Tabulator in drei Blöcke aufgeteilt. Der erste Block wird am ersten Tabulator ausgerichtet, der zweite Block am dritten Tabulator und der dritte Block am fünften Tabulator.

Möchte man ein Formelsystem an n Positionen ausrichten, so teilt man es in n Blöcke auf. Die Umgebungen `alignat` und `alignat*` bekommen die Zahl n als Argument. Man darf dann insgesamt $2n - 1$ Tabulatoren verwenden ($n - 1$ Tabulatoren zum Trennen der Blöcke und in jedem Block einen, um daran auszurichten).

Im obigen Beispiel sind die `{}` wichtig, damit die Abstände um $=$, $+$ und $-$ richtig gesetzt werden. Ohne `{}` erhielte man:

$$
\begin{alignedat}{3}
a &= & 3216b &+ & 13c & \\
17a &= & b &- & 2353c &
\end{alignedat}
$$

11.5 Formelnummern

Die Umgebungen `equation`, `gather`, `align` und `alignat` versehen jede Formel mit einer Formelnummer. Möchten Sie, daß eine bestimmte Formel nicht numeriert wird, so schreiben Sie

```
\notag
```

in den LaTeX-Code zur Erzeugung dieser Formel.

```
\begin{align}
  a^{2}+b^{2}&=c^{2}\notag\\
  a&<1
\end{align}
```

$$\begin{aligned} a^2 + b^2 &= c^2 \\ a &< 1 \qquad (8) \end{aligned}$$

Soll eine Formel nicht mit einer Nummer, sondern mit einer anderen Markierung (beispielsweise einem Stern) versehen werden, so schreiben Sie

`\tag{`⟨*Markierung*⟩`}`

in den LaTeX-Code zur Erzeugung dieser Formel.

```
\begin{equation}
  a^{2}+b^{2}=c^{2}\tag{$*$}
\end{equation}
```

$$a^2 + b^2 = c^2 \tag{$*$}$$

11.6 Referenzieren von Formeln

Mit Hilfe der Befehle

`\label{`⟨*Schlüssel*⟩`}`

und

`\eqref{`⟨*Schlüssel*⟩`}`

können Sie automatisch auf Formeln verweisen.

```
\begin{equation}
  a^{2}+b^{2}=c^{2}\tag{$\ast$}
  \label{eq:pythagoras}
\end{equation}
In \eqref{eq:pythagoras}
bezeichnen $a$ und $b$
die Katheten und $c$ die
Hypotenuse eines
rechtwinkligen Dreiecks.
```

$$a^2 + b^2 = c^2 \tag{$*$}$$

In ($*$) bezeichnen a und b die Katheten und c die Hypotenuse eines rechtwinkligen Dreiecks.

11.7 Brechen abgesetzter Formeln

Anders als eingebettete Formeln, kann $\LaTeX\,2_\varepsilon$ abgesetzte Formeln nicht selbsttätig brechen. Dies muß der/die Schreiber(in) des Textes selber tun. Das `amsmath`-Paket stellt ihm/ihr dazu die `split`-Umgebung zur Verfügung. Diese Umgebung funktioniert ähnlich wie die `align`-Umgebung: Die Stellen an denen eine Formel gebrochen werden soll, werden innerhalb der `split`-Umgebung durch den Befehl `\\` angegeben; und die entstehenden Formelzeilen werden an genau einer Position zueinander ausgerichtet, die durch das Tabulator-Zeichen `&` angegeben wird. Anders als die `align`-Umgebung kann die `split`-Umgebung jedoch nur innerhalb einer anderen Umgebung benutzt werden, die eine abgesetzte Formel oder ein Formelsystem erzeugt.

```
\begin{equation}
  \begin{split}
    f(x)&=a(a+1)x\\
        &=(a^{2}+a)x
  \end{split}
\end{equation}
```

$$\begin{split} f(x)&=a(a+1)x\\ &=(a^{2}+a)x \end{split} \tag{9}$$

Die `split`-Umgebung erzeugt selbst keine Formelnummer. Dies wird gegebenenfalls von der äußeren Umgebung (im Beispiel: `equation`) übernommen.

11.8 Satzzeichen in abgesetzten Formeln

Während man in eingebetteten Formeln keine Satzzeichen verwenden sollte, wird man dies in abgesetzten Formeln nicht umgehen können. Sie sollten dabei darauf achten, daß die Symbole, die Sie als Satzzeichen verwenden wollen, zur Klasse Punct gehören (Benutzen Sie also `\ldotp` statt `.` und `\colon` statt `:`).

11.9 Matrizen

Vom `amsmath`-Paket werden u. a. die folgenden Umgebungen zur Angabe einfacher Matrizen zur Verfügung gestellt: `matrix`, `pmatrix` und `vmatrix`. Mit diesen Umgebungen ist es möglich, Matrizen mit bis zu zehn Spalten und beliebig vielen Zeilen darzustellen.

Die Umgebungen unterscheiden sich in der Art der umschließenden Klammern, die sie erzeugen.

```
\begin{matrix}
  1 & 2\\
  3 & 4
\end{matrix}
```

$$\begin{matrix} 1 & 2\\ 3 & 4 \end{matrix}$$

```
\begin{pmatrix}
  1 & 2\\
  3 & 4
\end{pmatrix}
```

$$\begin{pmatrix} 1 & 2\\ 3 & 4 \end{pmatrix}$$

```
\begin{vmatrix}
  1 & 2\\
  3 & 4
\end{vmatrix}
```

$$\begin{vmatrix} 1 & 2\\ 3 & 4 \end{vmatrix}$$

Der Tabulator `&` trennt die einzelnen Einträge einer Zeile. Zum Trennen der einzelnen Zeilen dient (wie üblich) der Befehl `\\`.

Neben diesen Umgebungen gibt es noch die `smallmatrix`-Umgebung, die eine kleine Version der Matrix erzeugt, die man durch die `matrix`-Umgebung erhalten würde.

```
\begin{smallmatrix}
  1 & 2\\
  3 & 4
\end{smallmatrix}
```

$$\begin{smallmatrix} 1 & 2\\ 3 & 4 \end{smallmatrix}$$

Die Umgebungen `matrix`, `pmatrix`, `vmatrix` und `smallmatrix` funktionieren nur im mathematischen Modus.

Die einzelnen Einträge der Matrizen werden horizontal und vertikal zentriert. Sollen sie links- oder rechtsbündig gesetzt werden, müssen sie Ihnen ein `\hfill` nach- bzw. voranstellen:

```
\begin{pmatrix}
  1.1 & -1\\
  1   & 1
\end{pmatrix}
```

$$\begin{pmatrix} 1.1 & -1\\ 1 & 1 \end{pmatrix}$$

```
\begin{pmatrix}
  1.1      & -1\\
  1\hfill & \hfill 1
\end{pmatrix}
```

$$\begin{pmatrix} 1.1 & -1\\ 1\hfill & \hfill 1 \end{pmatrix}$$

Zum Kennzeichnen von Auslassungen dienen die Befehle `\hdots`, `\vdots` und `\ddots`:

```
\begin{pmatrix}
  1      & 0      & \hdots & 0      \\
  0      & \ddots & \ddots & \vdots\\
  \vdots & \ddots & \ddots & 0      \\
  0      & \hdots & 0      & 1
\end{pmatrix}
```

$$\begin{pmatrix} 1 & 0 & \hdots & 0 \\ 0 & \ddots & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0 \\ 0 & \hdots & 0 & 1 \end{pmatrix}$$

Blockmatrizen können Sie setzen, indem sie die gerade vorgestellten Umgebungen ineinander schachteln:

```
\begin{pmatrix}
  \begin{matrix}
    2 & 1\\
    0 & 2
  \end{matrix}   & 0\\
  0              & \begin{matrix}
                     3 & 1\\
                     0 & 3
                   \end{matrix}
\end{pmatrix}
```

$$\begin{pmatrix} \begin{matrix} 2 & 1\\ 0 & 2 \end{matrix} & 0\\ 0 & \begin{matrix} 3 & 1\\ 0 & 3 \end{matrix} \end{pmatrix}$$

11.10 Fallunterscheidungen

Für Fallunterscheidungen stellt Ihnen das `amsmath`-Paket die Umgebung `cases` zur Verfügung.

```
\begin{equation*}
  f(x)
  =
  \begin{cases}
    1, &\text{falls $x>0$}\\
    0  &\text{sonst}
  \end{cases}
\end{equation*}
```

$$f(x) = \begin{cases} 1, & \text{falls } x > 0\\ 0 & \text{sonst} \end{cases}$$

11.11 Definition weiterer Funktionen- und Operatornamen

Zusätzlich zu den in Tabelle 9 angegebenen langen Funktionen- und Operatornamen können Sie weitere Funktionen- und Operatornamen definieren. Am einfachsten funktioniert dies, wenn Sie das `amsmath`-Paket benutzen. Um einen neuen Befehl ⟨*Befehl*⟩ zu definieren, der den Funktionen- bzw. Operatornamen ⟨*Name*⟩ erzeugt, brauchen Sie dann nur

`\DeclareMathOperator{`⟨*Befehl*⟩**`}{`**⟨*Name*⟩**`}`**

in die Präambel der Eingabedatei zu schreiben.

```
\DeclareMathOperator{\arccot}{arccot}
...
\begin{equation*}
  \arccot^{\prime} x
  =
  -\frac{1}{1+x^{2}}
\end{equation*}
```

$$\operatorname{arccot}' x = -\frac{1}{1+x^2}$$

Bei einigen Operatoren sollen Hoch- und Tiefstellungen in abgesetzten Formeln nicht neben, sondern wie in

$$\lim_{n\to\infty}, \quad \max_{n=1}^{10}$$

über bzw. unter dem jeweiligen Operator erscheinen. Einen Befehl ⟨*Befehl*⟩, der einen derartigen Operatornamen ⟨*Name*⟩ erzeugt, können Sie mit dem Befehl

`\DeclareMathOperator*{`⟨*Befehl*⟩**`}{`**⟨*Name*⟩**`}`**

definieren.

```
\DeclareMathOperator*{\essinf}{essinf}
...
\begin{equation*}
  \essinf_{x>0}f(x)
\end{equation*}
```

$$\operatorname*{essinf}_{x>0} f(x)$$

Beachten Sie bitte, daß Sie die Befehle

`\DeclareMathOperator{`⟨*Befehl*⟩**`}{`**⟨*Name*⟩**`}`**

und

`\DeclareMathOperator*{`⟨*Befehl*⟩**`}{`**⟨*Name*⟩**`}`**

nur in der Präambel verwenden dürfen.

12 Abbildungen und Tabellen

In diesem letzten Abschnitt soll es um das „Beiwerk“ mathematischer Texte gehen: Abbildungen und Tabellen.

12.1 Einbinden von Bildern

Es ist möglich, in LaTeX 2ε-Texte Bilder einzubinden, die im „Encapsulated PostScript“-Format (also als `eps`-Datei) vorliegen. Am einfachsten geht dies mit dem `\includegraphics`-Befehl aus dem `graphicx`-Paket.

Um ein Bild einzubinden, laden Sie das `graphicx`-Paket und schreiben

`\includegraphics[`⟨*Skalierungsangaben*⟩`]{`⟨*Pfad*⟩`}`

an die Stelle der Eingabedatei, an der das Bild erscheinen soll. Das Argument ⟨*Pfad*⟩ enthält dabei den Pfad der Datei, in der das Bild gespeichert ist. Beachten Sie bitte, daß Sie als Trennzeichen in der Pfadangabe *nicht* den Backslash `\` verwenden dürfen, sodern stattdessen den Schrägstrich `/` benutzen müssen.

Mit Hilfe des optionalen Arguments ⟨*Skalierungsangaben*⟩ können Sie festlegen, wieviel Platz das Bild auf dem Papier einnehmen soll. Sie haben beispielsweise die Möglichkeit, die Bildbreite in der Form

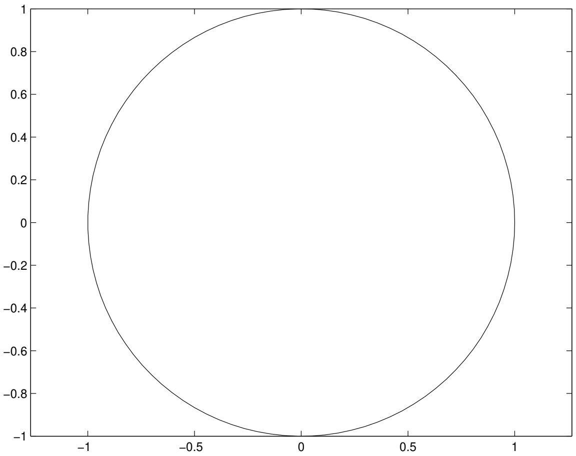
`width=`⟨*Breitenangabe*⟩

anzugeben (die Höhe wird dann automatisch angepaßt). Es empfiehlt sich, als Maßeinheit in ⟨*Breitenangabe*⟩ die aktuelle Zeilenbreite `\linewidth` zu verwenden.

Ein Beispiel: Wir wollen ein Bild einbinden, das unter dem Namen `kreis.eps` im aktuellen Verzeichnis gespeichert ist. Der Befehl

```
\includegraphics[width=0.7\linewidth]{kreis.eps}
```

ergibt dann:

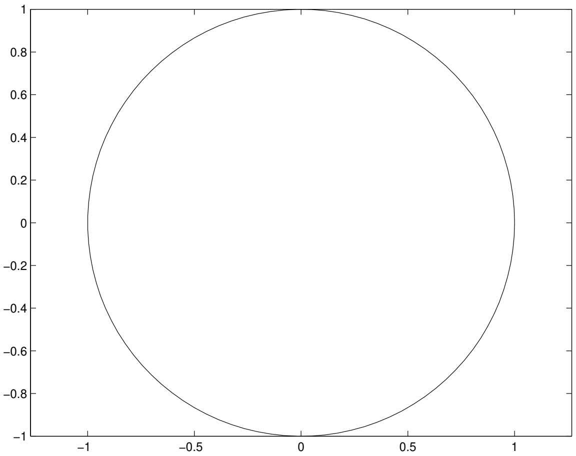


12.2 Zentrierungen

Im allgemeinen werden Sie Bilder oder Tabellen auf dem Papier horizontal zentrieren wollen. Hierzu können Sie die `center`-Umgebung verwenden. Indem Sie beispielsweise

```
\begin{center}
  \includegraphics[width=0.7\linewidth]{kreis.eps}
\end{center}
```

schreiben, erhalten Sie:



12.3 Tabellen

Zum Erzeugen „kurzer“ Tabellen, die auf eine einzige Seite passen, dienen in LaTeX 2_ε die Umgebungen `tabular` und `tabular*`. Wir wollen hier nur die `tabular`-Umgebung beschreiben. Erläuterungen zur `tabular*`-Umgebung finden Sie bei L. Lamport [9, Abschnitt C.10.2, S. 236].

Die `tabular`-Umgebung wird meistens in der Form

```
\begin{tabular}{⟨Format⟩}
  ⟨Zeilen⟩
\end{tabular}
```

verwendet.

Bei ⟨*Format*⟩ handelt es sich um eine Zeichenkette, die den Aufbau der einzelnen Zeilen angibt. Sie darf (unter anderem) die folgenden Zeichen enthalten:

`c` (steht für eine *zentrierte* Spalte),

`l` (steht für eine *links*bündige Spalte),

`r` (steht für eine *rechts*bündige Spalte),

`p{`⟨*Breite*⟩`}` (steht für einen Text der Breite ⟨*Breite*⟩),

`|` (steht für eine vertikale Linie).

Der Ausdruck ⟨*Zeilen*⟩ soll für die Zeilen der Tabelle stehen. Wie üblich wird zum Trennen der einzelnen Spalten einer Zeile der Tabulator `&` und zum Trennen der einzelnen Zeilen der Befehl `\\` benutzt. An den Anfang einer Zeile kann man den Befehl `\hline` schreiben, der eine horizontale Linie vor dieser Zeile erzeugt.

Als Beispiel geben wir eine Umrechnungstabelle für die Längeneinheiten Punkt (pt), Pica (pc), Inch (in) und Millimeter (mm) an:

```
\begin{tabular}{l|rrrr}
      &    pt &   pc &   in &    mm\\
  \hline
  1~pt &  1,00 & 0,08 & 0,01 &  0,35\\
  1~pc & 12,00 & 1,00 & 0,17 &  4,22\\
  1~in & 72,27 & 6,02 & 1,00 & 25,40\\
  1~mm &  2,85 & 0,24 & 0,04 &  1,00
\end{tabular}
```

	pt	pc	in	mm
1 pt	1,00	0,08	0,01	0,35
1 pc	12,00	1,00	0,17	4,22
1 in	72,27	6,02	1,00	25,40
1 mm	2,85	0,24	0,04	1,00

12.3.1 Text über mehrere Spalten

Zur Angabe von Text, der über mehrere Spalten reichen soll, dient der Befehl

`\multicolumn{`⟨*Anzahl*⟩`}{`⟨*Format*⟩`}{`⟨*Text*⟩`}`.

Das Argument ⟨*Anzahl*⟩ gibt die Anzahl der Spalten an, die durch ⟨*Text*⟩ überschrieben werden sollen. Das Argument ⟨*Format*⟩ gibt an, wie ⟨*Text*⟩ formatiert werden soll. Hier sind (unter anderem) wieder die Zeichen `c`, `l`, `r`, `p{`⟨*Breite*⟩`}` und `|` erlaubt.

Sie können den Befehl `\multicolumn` auch dazu benutzen, die Positionierung einzelner Einträge zu verändern.

```
\begin{tabular}{l|rrrr}
      & \multicolumn{3}{c}{nicht SI}
      & \multicolumn{1}{c}{SI}\\
      & \multicolumn{1}{c}{pt}
      & \multicolumn{1}{c}{pc}
      & \multicolumn{1}{c}{in}
      & \multicolumn{1}{c}{mm}\\
  \hline
  1~pt &  1,00 & 0,08 & 0,01 &  0,35\\
  1~pc & 12,00 & 1,00 & 0,17 &  4,22\\
  1~in & 72,27 & 6,02 & 1,00 & 25,40\\
  1~mm &  2,85 & 0,24 & 0,04 &  1,00
\end{tabular}
```

	nicht SI			SI
	pt	pc	in	mm
1 pt	1,00	0,08	0,01	0,35
1 pc	12,00	1,00	0,17	4,22
1 in	72,27	6,02	1,00	25,40
1 mm	2,85	0,24	0,04	1,00

12.4 Gleitobjekte

LaTeX 2ε verfügt über einen Mechanismus, mit dem Abbildungen und Tabellen automatisch so positioniert werden können, wie es aufgrund gewisser Formatierungsparameter am günstigsten erscheint. Die Abbildungen und Tabellen „gleiten“ dann an diese Stelle und werden deshalb als *Gleitobjekte* bezeichnet.

Abbildungen, die auf diese Weise positioniert werden sollen, werden in der Form

```
\begin{figure}[⟨Ort⟩]
  ⟨Abbildung⟩
  \caption{⟨Abbildungsunterschrift⟩}
\end{figure}
```

angegeben. Analog schreibt man für Tabellen:

```
\begin{table}[⟨Ort⟩]
  \caption{⟨Tabellenüberschrift⟩}
  ⟨Tabelle⟩
\end{table}.
```

Mit dem optionalen Argument ⟨*Ort*⟩ können Sie angeben, an welcher Stelle Sie das Gleitobjekt am liebsten eingefügt sehen würden. Das Argument ⟨*Ort*⟩ kann aus den Buchstaben `h`, `t`, `b`, `p` bestehen. Dabei bedeutet:

`h` hier (an der Stelle im Text, an der die `table`-Umgebung steht),

`t` oben (oben auf dieser oder einer der folgenden Seiten),

`b` unten (unten auf dieser oder einer der folgenden Seiten),

`p` auf einer Seite, auf der neben diesem Gleitobjekt höchstens weitere Gleitobjekte stehen.

Voreingestellt ist `tbp`, das heißt, LaTeX wird zunächst versuchen, das Gleitobjekt oben auf einer Seite auszugeben. Ist dies nicht möglich, wird LaTeX probieren, ob daß Objekt unten auf einer Seite ausgegeben werden kann. Wenn auch das nicht funktioniert, erscheint das Gleitobjekt auf einer Extraseite.

Das optionale Argument ⟨*Ort*⟩ ist dabei nicht viel mehr als ein Vorschlag. Wenn Sie beispielsweise nur `h` angeben, bedeutet das etwa: „Bitte, versuch doch, das Gleitobjekt hier unterzubringen.“ Sie können Ihrem Vorschlag mehr Nachdruck verleihen, indem Sie ihm ein `!` voranstellen. Schreiben Sie beispielsweise `!h`, so bedeutet das soviel wie: „Versuch doch, das Gleitobjekt hier unterzubringen, selbst wenn dir das nicht optimal erscheint.“

Die Numerierung der Gleitobjekte erfolgt automatisch. Es ist auch wieder möglich, mit Hilfe der Befehle `\label{`⟨*Schlüssel*⟩`}` und `\ref{`⟨*Schlüssel*⟩`}` automatisch auf Gleitobjekte zu verweisen. Dazu sollten Sie den `\label`-Befehl direkt hinter den `\caption`-Befehl schreiben.

```
In Abbildung~\ref{fig:kreis}
zeichnen wir einen Kreis.
\begin{figure}
  \begin{center}
    \includegraphics[width=0.7\linewidth]{%
      kreis.eps}
  \end{center}
  \caption{Ein Kreis mit Radius eins}%
    \label{fig:kreis}
\end{figure}
```

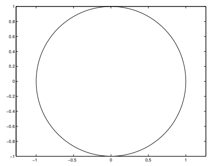


Abbildung 1: Ein Kreis mit Radius eins.

In Abbildung 1 zeichnen wir einen Kreis.

Wenn Sie die Bezeichnung „Abbildung“ oder „Tabelle“ in den Unter- bzw. Überschriften nicht für passend halten, verändern sie einfach in der Präambel den Befehl `\figurename` bzw. `\tablename`.

12.5 Abbildungs- und Tabellenverzeichnis

Mit den Befehlen `\listoffigures` und `\listoftables` können Sie ein Abbildungs- bzw. Tabellenverzeichnis erstellen. In diese werden die jeweiligen Abbildungsunter- bzw. Tabellenüberschriften, die Sie in den einzelnen `\caption`-Befehlen angegeben haben, übernommen.

Manchmal bietet es sich an, anstelle einer besonders langen Abbildungsunter- oder Tabellenüberschrift eine prägnantere Kurzform in das Abbildungs- bzw. Tabellenverzeichnis einzutragen. Eine solche Kurzform können Sie als optionales Argument im `\caption`-Befehl angeben. Schreiben Sie also beispielsweise innerhalb einer `figure`-Umgebung

```
\caption[Ein Kreis]{Ein Kreis mit Radius eins},
```

so wird anstelle der Bildunterschrift „Ein Kreis mit Radius eins" nur der Text „Ein Kreis" in das Abbildungsverzeichnis eingetragen.

Die Überschrift des Abbildungs- bzw. Tabellenverzeichnisses können Sie verändern, indem Sie den Befehl `\listfigurename` bzw. `\listtablename` umdefinieren.

Abbildungs- und Tabellenverzeichnisse werden von LaTeX auf die gleiche Weise wie das Inhaltsverzeichnis erzeugt. Ihre Einträge sind also frühestens nach dem zweiten LaTeX-Lauf korrekt. Es empfiehlt sich, LaTeX nach der letzten Änderung des `tex`-Files *dreimal* durchlaufen zu lassen.

Literatur

[1] American Mathematical Society. *$\mathcal{AMS}$-LaTeX Version 1.2 User's Guide.* November 1996.

[2] D. P. Carlisle und S. P. Q. Rahtz. *The graphicx package.* Juni 1995.

[3] Michel Goossens, Frank Mittelbach und Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, Reading (Mass.), 1994.

[4] Michel Goossens, Frank Mittelbach und Alexander Samarin. *Der LaTeX-Begleiter.* Addison-Wesley, Bonn, 1995.

[5] Nicholas J. Higham. *Handbook of Writing for the Mathematical Sciences.* SIAM, Philadelphia, 1993.

[6] Jörg Knappen, Hubert Partl, Elisabeth Schlegl und Irene Hyna. *LaTeX 2$_\varepsilon$-Kurzbeschreibung.* 1994.

[7] Donald E. Knuth. *The TeXbook.* Addison-Wesley, Reading (Mass.), 17. Aufl., 1990.

[8] Leslie Lamport. *LaTeX: A document Preparation System.* Addison-Wesley, Reading (Mass.), 2. Aufl., 1994.

[9] Leslie Lamport. *Das LaTeX-Handbuch.* Addison-Wesley, Bonn, 1995.