LaTeX 実習マニュアル

第零版 revision02
2008 年 3 月 12 日（水）

目次

第 1 章　LaTeX の基礎

1.1　LaTeX とは何か

1.1.1　組版

この「LaTeX 実習マニュアル」という文章は、LaTeX というものについて解説することを目的とするチュートリアルです（LaTeX は、「ラテフ」または「ラテック」と読みます）。

活字などを組み合わせることによって文書の原稿から印刷物のページを作るという作業のことを「組版」(typesetting) と呼びます。この文章が解説している LaTeX というのは、Leslie Lamport さんという人によって作られた、組版を実行するソフトウェアのことです。LaTeX を使うことによって、美しく組版された印刷物を作ることができます。

1.1.2　TeX

LaTeX というのは、Donald Knuth さんという人によって作られた、TeX という組版のソフトウェアを基盤として作られたものです（TeX は、「テフ」または「テック」と読みます）。

TeX と LaTeX との相違点は、前者は文書の物理構造を扱い、後者は文書の論理構造を扱う、というところにあります。物理構造 (physical structure) というのは文字の大きさや書体やレイアウトなどのことで、論理構造 (logical structure) というのは見出しや引用文や注釈などのことです。つまり、TeX を使う人は、文字の大きさや書体やレイアウトなどを指定することによって組版を実行させるのに対して、LaTeX を使う人は、見出しや引用文や注釈などを指定することによって組版を実行させるということです。

ただし、LaTeX では文書の物理構造を扱うことが不可能だ、というわけではありません。LaTeX でも、文字の大きさや書体やレイアウトなどを指定することは可能です。

1.1.3　マークアップ言語

文書が持っている構造を記述するために文書の中に埋め込まれる文字列は、「マークアップ」(markup) と呼ばれます。そして、マークアップを記述するために使われる言語は、「マークアップ言語」(markup language) と呼ばれます。たとえば、ウェブページを記述するために使われる HTML というのは、マークアップ言語の一種です。

TeX に原稿の組版を実行させるためには、その原稿にマークアップを埋め込む必要があります。TeX は、マークアップが埋め込まれた原稿を読み込んで、それを処理して、組版の結果を出力します。LaTeX のような、TeX を基盤として作られたソフトウェアも同様です。

TeX や LaTeX などのソフトウェアの名前は、それらのソフトウェアが理解することのできるマークアップ言語の名前としても使われます。

なお、この文章の中では、LaTeX のマークアップが埋め込まれた原稿のことを、「LaTeX ソース」と呼ぶことにします。

1.1.4　TeX と LaTeX のロゴ

TeX や LaTeX などの名前は、通常、このようなロゴを使って表記されます。このロゴは、TeX や LaTeX などを使っている場合には簡単に出力することができるのですが、場合によってはロゴを使うことができないこともあります。そのような場合は、TeX は TeX、LaTeX は LaTeX というように、大文字と小文字を組み合わせた形で名前を表記するという慣習になっています。

1.2　LaTeX の使い方

1.2.1　LaTeX ソースの入力

文書の原稿を LaTeX に組版させたいときは、まず、原稿に LaTeX のマークアップを埋め込んだもの、つまり LaTeX ソースを作る必要があります。LaTeX ソースというのは単なるテキストデータですので、テキストエディターを使うことによって、それを入力してファイルに保存することができます。

それでは、何らかのテキストエディターを使って次の LaTeX ソースを入力して、`greet.tex` という名前のファイルにそれを保存してください。

LaTeX ソースの例　`greet.tex`

```
\documentclass{jarticle}
\begin{document}
こんにちは。
\end{document}
```

ちなみに、LaTeX ソースを格納するファイルの名前には、このように .tex という拡張子を付けます。

1.2.2 LaTeX ソースのコンパイル

LaTeX が、LaTeX ソースを読み込んで、それを処理して、組版の結果を出力することを、LaTeX ソースを「コンパイルする」(compile) と言います。

LaTeX を起動するコマンドは、

platex パス名

と書きます。この中のパス名で、LaTeX ソースを指定します（拡張子の .tex は省略することもできます)。LaTeX は、指定されたファイルに格納されている LaTeX ソースを処理して、それを組版した結果をカレントディレクトリのファイルに出力します。出力先のファイルの名前は、LaTeX ソースのファイル名の拡張子を .dvi に変更したものになります。

それでは、実際に試してみましょう。先ほど入力した LaTeX ソースのファイルのあるディレクトリをカレントディレクトリにして、

```
platex greet
```

というコマンドをシェルに入力してください。すると、greet.tex に格納されている LaTeX ソースが処理されて、エラーがなければ、組版の結果が greet.dvi というファイルに出力されます。

LaTeX が組版の結果を出力するファイルは、「dvi ファイル」(dvi file) と呼ばれます。dvi という言葉は、「特定の機器に依存しない」という意味を持つ、device independent という言葉を縮めたものです。

1.2.3 dvi ファイルの表示と印刷

次に、LaTeX による組版の結果をモニターの画面で確認してみましょう。

dvi ファイルに保存された組版の結果をモニターの画面に表示したり、それをプリンターで印刷したりするソフトウェアは、「プレビューア」(previewer) と呼ばれます。プレビューアとしては、Linux や MacOS では xdvi、Windows では dviout がもっともポピュラーです。

プレビューアの使い方については、それぞれのプレビューアに付属している文書を参照してください。

1.2.4 dvi ファイルから PDF への変換

組版の結果をメールに添付して誰かに送ったり、ウェブで配布したりするときには、通常、dvi ファイルに出力された結果を PDF に変換します。「PDF」というのは、Portable Document Format という言葉から作られた頭字語で、印刷物のデータを交換するための標準的なデータ形式の名前です。

dvi ファイルの内容を PDF に変換する方法には、いくつかのものがあります。そのうちのひとつは、dvipdfm（あるいはそれを拡張した dvipdfmx）というソフトウェアを使うという方法です。dvipdfm を起動するコマンドは、

dvipdfm パス名

と書きます。この中のパス名で、dvi ファイルを指定します（拡張子の .dvi は省略することもできます)。dvipdfm は、指定されたファイルに格納されている組版の結果を PDF に変換して、その結果をファイルに出力します。出力先のファイルの名前は、デフォルトでは、dvi ファイルの拡張子を .pdf に変更したものになります。

それでは、実際に試してみましょう。

```
dvipdfm greet
```

というコマンドをシェルに入力してください。すると、greet.dvi に格納されている組版の結果が PDF に変換されて、その結果が greet.pdf というファイルに出力されます。

1.3 LaTeX の基礎の基礎

1.3.1 コマンド

LaTeX ソースの中で使われるマークアップは、「コマンド」(command) と呼ばれます。たとえば、前の節で入力した LaTeX ソースの中には、

```
\documentclass{jarticle}
\begin{document}
\end{document}
```

という 3 個のコマンドが含まれています。

1.3.2 コマンド名

コマンドは、かならずコマンド名を含んでいます。

「コマンド名」(command name) というのはコマンドの種類を識別するための名前のことです。大多数のコマンド名は、バックスラッシュ(backslash, \) で始まります（ソフトの環境によっては、バックスラッシュが円マーク (¥) で表示されることもありますが、どちらで表示されても問題はありません)。前の節の LaTeX ソースに含まれているコマンドの場合は、

```
\documentclass    \begin    \end
```

というのがコマンド名です。

コマンド名の形式には、次の三つの種類があります。

(1) バックスラッシュの右側に英字などを何文字か並べたもの。
(2) バックスラッシュの右側に英字以外の文字を 1 個だけ並べたもの。
(3) 1 個の特殊文字だけから構成されるもの。

大多数のコマンド名は 1 番目の形式で、2 番目と 3 番目の形式を持つコマンド名は、きわめて少数です。

何々というコマンド名で種類が指定されるコマンドは、「何々コマンド」と呼ばれます。たとえば、`\begin` というコマンド名で種類が指定されるコマンドは、「`\begin` コマンド」と呼ばれます。

1.3.3 引数

コマンドの種類によっては、コマンド名の右側に何個かの引数を書く必要があります。

「引数」(argument) というのは、コマンドの意味を具体的に決定する文字列のことです。引数は、コマンド名の右側に、中括弧 (`{}`) で囲んで書きます。前の節の LaTeX ソースに含まれているコマンドの場合は、

```
{jarticle}   {document}
```

というのが引数です。

1.3.4 環境

LaTeX では、文書の中の範囲で、何らかの意味付けが与えられているもののことを、「環境」(environment) と呼びます。そして、環境の種類をあらわす名前は、「環境名」(environment name) と呼ばれます。

環境は、`\begin` と`\end` という二つのコマンドを書くことによって作ることができます。`\begin` は環境を開始させるコマンドで、`\end` は環境を終了させるコマンドです。ですから、まず`\begin` コマンドを書いて、それよりも後ろに`\end` コマンドを書くと、それらのコマンドで囲まれた範囲が環境になります。環境の種類は、環境名をそれらのコマンドの引数として書くことによって指定します。

前の節の LaTeX ソースに含まれている、

```
\begin{document}
\end{document}
```

というコマンドの組は、`document` という名前で指定される種類の環境を作っています。

何々という環境名で種類が指定される環境は、「何々環境」と呼ばれます。たとえば、document という環境名で種類が指定される環境は、「document 環境」と呼ばれます。

1.3.5 document 環境

document 環境は、その名前のとおり、「ここは文書である」ということを意味する環境です。LaTeX ソースは、かならず、document 環境を 1 個だけ含んでいないといけません。

LaTeX が組版の対象とするのは、document 環境の中に書かれているものだけです。その外側に書かれているものは、組版の対象とはなりません。

1.3.6 プリアンブル

LaTeX ソースの先頭から、

```
\begin{document}
```

というコマンドの直前までの部分は、「プリアンブル」(preamble) と呼ばれます。プリアンブルは、コマンドや環境の定義を変更したり、新しいコマンドや環境を定義したりする記述を書くための場所です。

1.3.7 文書クラス

LaTeX ソースの先頭には、かならず、\documentclass というコマンドを 1 個だけ書かないといけません。このコマンドは、文書クラスというものを指定するためのものです。「文書クラス」(document class) というのは、文書の種類ごとに定義されている組版のスタイルのことです。

\documentclass コマンドには、引数として、指定する文書クラスの名前を書く必要があります。日本語の文書を作る場合の文書クラスとしては、次のようなものがあります。

jarticle	比較的短い文書のための文書クラス。
jreport	いくつかの章から構成される、比較的長い文書のための文書クラス。ページの左右は同じレイアウト。
jbook	書籍のための文書クラス。レイアウトがページの左右で異なる。

前の節で入力した LaTeX ソースの先頭には、

```
\documentclass{jarticle}
```

というコマンドが書かれています。これは、文書クラスとして jarticle を指定する、という意味です。

1.3.8 オプション引数

通常の引数は、コマンドの種類ごとに必要な個数が決まっているのですが、それとは別に、書けばその指定が反映されて、書かなければデフォルトの意味になる、という特殊な引数もあります。そのような引数は、「オプション引数」(optional argument) と呼ばれます。

たとえば、\documentclass コマンドは、コマンド名と引数とのあいだにオプション引数を書くことによって、文書を出力する紙の大きさや、本文の文字の大きさなどを設定することができます。

オプション引数は、中括弧ではなくて、角括弧 ([]) で囲んで書きます。

1.3.9 紙の大きさ

文書を出力する紙の大きさは、デフォルトでは A4 が設定されていますが、オプション引数の中に a5paper と書けば A5 が設定されて、b4paper と書けば B4 が設定されて、b5paper と書けば B5 が設定されます。たとえば、

```
\documentclass[b5paper]{jarticle}
```

と書くことによって、紙の大きさとして B5 を設定することができます。

1.3.10 本文の文字の大きさ

本文の文字の大きさは、デフォルトでは 10 ポイントが設定されていますが、オプション引数の中に 11pt と書けば 11 ポイントが設定されて、12pt と書けば 12 ポイントが設定されます。たとえば、

```
\documentclass[12pt]{jarticle}
```

と書くことによって、本文の文字の大きさとして 12 ポイントを設定することができます。

紙の大きさと本文の文字の大きさを両方とも設定したいときは、

```
\documentclass[b4paper,11pt]{jarticle}
```

というように、それらを意味する文字列をコンマで区切って並べます（順番は逆でもかまいません）。

1.3.11 注釈

LaTeX ソースの中に書かれている文字列の中で、LaTeX がそれを無視するもののことを、「注釈」(comment) と呼びます。

LaTeX ソースの中に注釈を書きたいときは、パーセント (%) という文字を使います。LaTeX ソースの中に % を書くと、その直後から最初の改行までが注釈とみなされます。

LaTeX ソースの例 comment.tex

```
\documentclass{jarticle}
\begin{document}
% ここは注釈です。
こんにちは。% ここも注釈です。
\end{document}
```

1.3.12 ロゴ

TeX と LaTeX というロゴは、そのためのコマンドを書くことによって、簡単に出力することができます。TeX を出力するコマンドは\TeX、LaTeX を出力するコマンドは\LaTeX で、いずれも引数は不要です。

なお、バックスラッシュの右側に何文字かの英字などを並べたコマンド名を持つコマンドで、引数のないものを書く場合は、その末尾に半角の空白などを書くことによって、コマンドの終わりを明示する必要があります。

LaTeX ソースの例 logo.tex

```
\documentclass{jarticle}
\begin{document}
\LaTeX は、\TeX を基盤として作られています。
\end{document}
```

1.4 文字

1.4.1 文字の基本

document 環境の中に書かれた文字は、基本的には、組版の結果にそのまま出力されます。しかし、そのとおりには出力されない文字もあります。たとえば、コマンドに含まれる文字はそのとおりに出力されるとは限りませんし、注釈に含まれる文字はまったく出力されません。

コマンドや注釈の中で使われている場合を除いて、半角の英字、数字、特殊文字の一部は、そのとおりに出力されます。全角文字も、そのとおりに出力されます。しかし、半角の特殊文字のうちで、

```
# $ % & _ { } ~ ^ \ | < >
```

という 13 個の文字は、そのとおりの出力にはなりません。

1.4.2 空白文字

LaTeX では、半角の空白、タブ、改行という 3 種類の文字を総称して、「空白文字」(space character) と呼びます。連続する何個かの空白文字は、1 個の空白として出力されます。

全角の空白は、そのまま全角の空白として出力されます。

改行は、半角文字の直後にある場合は 1 個の空白文字として扱われますが、全角文字の直後にある場合は完全に無視されます。ですから、LaTeX で日本語の文書を作る場合は、全角文字の直後であれば、どこにでも好きなところに改行を入れることができます。

1.4.3 段落

空白文字だけしか含まれていない行のことを「空行」(empty line) と呼びます。

LaTeX は、1 個以上の連続する空行を、段落 (paragraph) の終わりという意味だと解釈します。ですから、空行を入力することによって、段落を終わらせて次の段落を始めることができます。

LaTeX ソースの例 danraku.tex

```
\documentclass{jarticle}
\begin{document}
日本語の文章は、普通に書いていけばほとんどそのとおりに
組版されます。ただし、改行は組版には反映されません。

改行を二つ入力して空行を作ると、そこで段落が終わって、
新しい段落が始まります。
\end{document}
```

1.4.4 強制的な改行

段落を終わらせるのではなくて、段落の途中で強制的に改行させたいときは、\\というコマンドを書きます。

LaTeX ソースの例 newline.tex

```
\documentclass{jarticle}
\begin{document}
この文は、途中で強制的に\\改行させられています。
\end{document}
```

1.4.5 そのとおりには出力されない文字の出力

この節の最初のところで紹介したように、LaTeX には、そのとおりの出力にならない 13 個の文字があります。それらの文字のうちで、

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

という 7 個については、

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

というように、バックスラッシュの右側にその文字を並べたコマンド名を持つコマンドを書くことによって出力することができます。~（チルダ）と^（サーカムフレックス）は、

```
\~{}  \^{}
```

というように、バックスラッシュの右側にその文字を並べたコマンド名のさらに右側に空の引数を書くことによって出力することができます。< と > と | は、

```
$<$  $>$  $|$
```

というように、その文字をドルマークで囲むことによって出力することができます（この場合のドルマークの意味については、第 2.1 節で説明します）。\は、

```
$\backslash$
```

と書くことによって出力することができます。

LaTeX ソースの例 tokushu.tex

```
\documentclass{jarticle}
\begin{document}
\# \$ \% \& \_ \{ \} \~{} \^{} $<$ $>$ $|$ $\backslash$
\end{document}
```

1.4.6 任意の文字をそのとおりに出力する方法

LaTeX には、任意の文字をそのとおりに出力するためのコマンドと環境があります。そのとおりに出力されない文字を含む文字列をそのとおりに出力したいときや、空白や改行によって整形された文字列をその形のとおりに出力したいときなどには、そのコマンドまたは環境が役に立ちます。

\verb は、引数をそのとおりに出力するコマンドで、 verbatim は、その中の文字列をそのとおりに出力する環境です。このどちらかを使うことによって、通常ならばそのとおりには出力さ

れない文字を、そのとおりに出力することができます。

\verb の引数の書き方は、普通のコマンドとは大きく異なっています。普通のコマンドの場合は、引数を中括弧 ({}) で囲むのですが、\verb コマンドの場合は、引数を任意の同じ文字（ただし、空白、英字、アスタリスク (*) は不可）で囲みます。たとえば、\verb|{<>}| というコマンドは{<>}を出力し、\verb+#$%&+というコマンドは#$%&を出力し、\verb3_|~3 というコマンドは_|~を出力します。

\verb コマンドと verbatim 環境のそれぞれには、\verb*コマンドと verbatim* 環境という、類似した機能を持つ兄弟のようなコマンドと環境があります。アスタリスクが付いているほうは、半角の空白が␣という記号で出力されるという点を除いて、アスタリスクのないほうと同じ動作をします。

なお、\verb コマンド、\verb*コマンド、 verbatim 環境、 verbatim* 環境によって出力される半角の文字には、 typewriter という書体が使われます。

LaTeX ソースの例 verb.tex

```
\documentclass{jarticle}
\begin{document}
\verb/# $ % &/
\verb*/# $ % &/

\begin{verbatim}
#  $  %  &  _  {  }  ~  ^  \  |  <  >
\end{verbatim}
\begin{verbatim*}
#  $  %  &  _  {  }  ~  ^  \  |  <  >
\end{verbatim*}
\end{document}
```

1.4.7 引用符

英語の文章で使われる引用符には、単一引用符と二重引用符の 2 種類があります。また、それぞれに左側用と右側用があります。

左側の単一引用符は、逆引用符 (`) という文字を書くことによって出力させることができます。右側の単一引用符は、普通の単一引用符 (') を書けば、それがそのとおりには出力されます。

逆引用符を二つ連続して書くと、それは左側の二重引用符として出力されます。同じように、単一引用符を二つ連続して書くと、それは右側の二重引用符として出力されます。

LaTeX ソースの例 quote.tex

```
\documentclass{jarticle}
\begin{document}
We can use `single' and ``double'' quotation marks.
\end{document}
```

1.4.8 ダッシュ

LaTeX ソースの中にマイナス (-) という文字を書くと、それは、単語の内部を区切るためのハイフンとして出力されます。

二つのマイナスを連続して書いたもの (--) は、数字の範囲を示す場合などに使われるダッシュを出力する命令です。

三つのマイナスを連続して書いたもの (---) は、文章の区切りとして使われるダッシュを出力する命令です。

LaTeX ソースの例 dash.tex

```
\documentclass{jarticle}
\begin{document}
She is my sister-in-law.
This table shows my income during 2001--2007.
This is a punctuation dash---I think.
\end{document}
```

1.4.9 空白の出力

LaTeX は、ピリオド (.)、疑問符 (?)、感嘆符 (!)、コロン (:) の後ろに空白文字が書かれている場合、その場所を文末とみなして、通常よりも少し広いスペースを出力します。しかし、たとえば etc. のように、文末ではないにもかかわらずピリオドなどを書く必要がある場合もあります。そのような場合は、そこが文末ではないということを LaTeX に示すことが必要になります。

ピリオドなどが文末ではないということを LaTeX に示したいときは、通常、空白を出力するコマンドというのを使います。空白を出力するコマンドというのは、\␣というコマンドのことです（␣のところには、1 個の空白を書きます）。

空白を出力するコマンドは、たとえば、\LaTeX というコマンドによって出力されるロゴの後ろに空白を出力したいというような、本来ならば出力されない空白を出力したいときにも使われます。

空白を出力するコマンドとしては、\␣のほかに、~（チルダ）というコマンドもあります。このコマンドは、その位置で行を上下に分けないようにして空白を出力する、ということを LaTeX に指示します。たとえば、Chapter~2 と書くことによって、Chapter と 2 とが上下の行に分かれるのを防ぐことができます。

LaTeX ソースの例 space.tex

```
\documentclass{jarticle}
\begin{document}
Mr.~Tanaka et al.\ are my friends.
They told me that \LaTeX\ was wonderful.
\end{document}
```

1.4.10 文字に付ける記号

文字に対してアクサン (accent) やウムラウト (Umlaut) などの記号を付けたいときは、次のようなコマンドを使います。

\'{o} ó　\`{o} ò　\^{o} ô　\"{o} ö　\~{o} õ　\={o} ō

i または j の上に記号を付ける場合は、それらの本来の形ではなくて、本来の形から点を取り除いた、ɪ や ȷ という形の文字を使う必要があります。これらの点のない文字は、\i と\j というコマンドを書くことによって出力することができます。

LaTeX ソースの例 accent.tex

```
\documentclass{jarticle}
\begin{document}
\'{o} \`{o} \^{o} \"{o} \~{o} \={o} \={i} \={j} \={\i} \={\j}
\end{document}
```

1.5 文書の物理構造

1.5.1 この節について

第 1.1 節で説明したように、LaTeX を使う人は、見出しや引用文や注釈などの論理構造を指定することによって組版を実行させるわけですが、文字の大きさや書体やレイアウトなどの物理構造を LaTeX で指定することが不可能というわけではありません。

この節では、LaTeX ソースの中で文書の物理構造を指定する方法について説明したいと思います。

1.5.2 文字の大きさ

文字の大きさは、次のコマンドを使うことによって指定することができます。

\tiny	文字 letter
\scriptsize	文字 letter
\footnotesize	文字 letter
\small	文字 letter
\normalsize	文字 letter
\large	文字 letter

\Large 文字 letter
\LARGE 文字 letter
\huge 文字 letter
\Huge 文字 letter

これらのコマンドで文字の大きさが指定されなかった場合は、\normalsize コマンドで指定される大きさになります。

1.5.3 宣言

文字の大きさを指定するコマンドは、それが書かれている位置よりも後ろに書かれている文字の大きさに影響を与えます。そのような、それが書かれている位置よりも後ろに書かれているものに影響を与えるコマンドは、「宣言」(declaration) と呼ばれます。

宣言の影響は、その宣言を囲んでいる中括弧 ({}) が閉じる場所、またはその宣言を含んでいる環境が終わる場所まで続きます。たとえば、

```
吾輩は{\Large 猫}である
```

と書いたとすると、

吾輩は猫である

と出力されます。

LaTeX ソースの例 letsize.tex

```
\documentclass{jarticle}
\begin{document}
{\tiny 小}{\scriptsize 小}{\footnotesize 小}{\small 小}
普通{\large 大}{\Large 大}{\LARGE 大}{\huge 大}{\Huge 大}
\end{document}
```

1.5.4 日本語の文字の書体

次に、書体 (font) を指定するコマンドを紹介しましょう。

日本語の平仮名、片仮名、漢字などの書体としては、通常、明朝体またはゴシック体が使われます。これらの二つの書体は、次のコマンドによって指定されます。

\textmc 明朝体
\textgt **ゴシック体**

これらのコマンドは、指定された書体で引数を出力します。これらのコマンドによって書体が指定されなかった場合、日本語の文字は明朝体で出力されます。ですから、\textmc というコマンドが必要になることは、めったにありません。

LaTeX ソースの例 japanese.tex

```
\documentclass{jarticle}
\begin{document}
明朝体と\textgt{ゴシック体}。
\end{document}
```

1.5.5 ラテン文字の書体

イタリア語、フランス語、スペイン語、ドイツ語、英語などで使われている文字は、「ラテン文字」(Latin letter) と呼ばれます。ラテン文字の書体は、ファミリー (family)、シェイプ (shape)、シリーズ (series) という三つの要素によって決定されます。

ファミリーというのは書体のデザインのことで、次のコマンドによって指定されます。

\textrm roman
\textsf sans serif
\texttt typewriter

シェイプというのは書体のバリエーションのことで、次のコマンドによって指定されます。

\textup upright
\textit *italic*
\textsl *slanted*
\textsc Small Caps

シリーズというのは文字の線の太さのことで、次のコマンドによって指定されます。

\textmd medium
\textbf **boldface**

これらのコマンドも、日本語の文字の書体を指定するコマンドと同じように、指定された書体で引数を出力します。ファミリーとシェイプとシリーズの組み合わせは、これらのコマンドを入れ子にすることによって指定されます。たとえば、

```
\texttt{\textsc{Hello, world!}}
```

というコマンドを書くことによって、ファミリーとして typewriter、シェイプとして Small Caps を指定して、

HELLO, WORLD!

と出力することができます。ただし、すべての組み合わせについて、期待したとおりの書体で出力されるとは限りません。

なお、これらのコマンドによって書体が指定されなかった場合、ファミリーは roman、シェイプは upright、シリーズは medium になります。

LaTeX ソースの例 latin.tex

```
\documentclass{jarticle}
\begin{document}
[roman upright medium],
[\textbf{roman upright boldface}],
[\textit{roman italic medium}],
[\textit{\textbf{roman italic boldface}}],
[\textsl{roman slanted medium}],
[\textsc{Roman Small Caps Medium}],
[\textsf{sans serif upright medium}],
[\textsf{\textit{sans serif italic medium}}],
[\texttt{typewriter upright medium}],
[\texttt{\textit{typewriter italic medium}}],
[\texttt{\textsl{typewriter slanted medium}}].
\end{document}
```

1.5.6 左寄せと右寄せとセンタリング

次の環境を使うことによって、文字列を左に寄せたり、右に寄せたり、センタリングしたりすることができます。

`flushleft` 左寄せ。
`flushright` 右寄せ。
`center` センタリング。

\\コマンドを使うことによって、これらの環境の中で改行を出力することも可能です。

LaTeX ソースの例 align.tex

```
\documentclass{jarticle}
\begin{document}
\begin{flushleft}
在校生の皆様へ
\end{flushleft}
\begin{flushright}
大阪市天王寺区勝山 4 丁目 5 番 6 号\\
大阪魔法学校
\end{flushright}
\begin{center}
\textgt{\Large 占星術講習会のご案内}
\end{center}
```

```
夏休みに占星術の講習会を開催しますので、ぜひご参加ください。
\end{document}
```

1.6 文書の論理構造

1.6.1 タイトル

この節では、文章の論理構造を指定する各種のコマンドや環境を紹介していきたいと思います。

最初に紹介する論理構造は、タイトルです。

文書のタイトルは、\maketitle というコマンドを使うことによって出力することができます。

\maketitle は、あらかじめ定義されているタイトルと著者名と日付を出力するコマンドです。したがって、このコマンドでタイトルなどを出力するためには、このコマンドが書かれている場所よりも上のところで、タイトルなどを定義する必要があります。

タイトルと著者名と日付は、次のコマンドによって定義されます。

\title	タイトル
\author	著者名
\date	日付

LaTeX ソースの例 title.tex

```
\documentclass{jarticle}
\begin{document}
\title{文書のタイトル}
\author{著者名}
\date{2007 年 12 月 1 日}
\maketitle
ここからが本文です。
\end{document}
```

1.6.2 タイトルに関連するコマンド

タイトルと著者名と日付を定義するコマンドの引数の中では、次のようなコマンドを使うことができます。

\\	改行を出力します。
\and	複数の著者名を列挙するとき、それぞれの著者名をこのコマンドで区切ります。
\thanks	このコマンドの引数が脚注になります。通常、著者の所属組織を注記するために使われます。

LaTeX ソースの例 title2.tex

```
\documentclass{jarticle}
\begin{document}
\title{文書のタイトルを\\複数の行に\\分ける方法について}
\author{著者一\thanks{梅田大学} \and 著者二\thanks{京橋大学}
\and 著者三\thanks{鶴橋大学} \and 著者四\thanks{桃谷大学}}
\date{2007 年 12 月 1 日}
\maketitle
\end{document}
```

1.6.3 セクション単位

文章を階層的に構成するそれぞれの部分は、「セクション単位」(sectional unit) と呼ばれます。たとえば、部 (part)、章 (chapter)、節 (section)、項 (subsection, article) などのセクション単位があります。

LaTeX では、セクション単位は、「セクショニングコマンド」(sectioning command) と呼ばれるコマンドを書くことによって記述されます。

セクショニングコマンドには、次のようなものがあります。

\part	部
\chapter	章

`\section` 節
`\subsection` 項

セクショニングコマンドは、それぞれのセクション単位の先頭に書きます。そして、これらのコマンドには、1 個の引数を書く必要があります。この引数は、セクション単位の先頭に見出し (heading) として出力されます。

セクション単位の見出しの先頭には、通常、そのセクション単位の番号を示す数字が出力されます。セクション単位の番号は、最初はゼロになっていて、セクショニングコマンドによってそれに 1 が加算されます。ですから、セクション単位の番号は、1, 2, 3, ···, と自動的に増加していきます。

また、上位のセクション単位のセクショニングコマンドは、下位のセクション単位の番号をゼロにリセットします。たとえば、節のセクショニングコマンドは、項の番号をゼロにリセットします。ただし、`\part` コマンドは、下位のセクション単位の番号に影響を与えません。

`chapter` というセクショニングコマンドは、文書クラスが `jarticle` の場合は使えません。その理由は、文書クラスが `jarticle` の文書をいくつか集めて、 `jreport` または `jbook` の文書を作る場合、 `jarticle` の文書をひとつの章として扱うことができるようにするためです。

LaTeX ソースの例 `section.tex`

```
\documentclass{jarticle}
\begin{document}
\section{一番目の節}
\subsection{一番目の節の一番目の項}
一番目の節の一番目の項の本文。
\subsection{一番目の節の二番目の項}
一番目の節の二番目の項の本文。
\section{二番目の節}
\subsection{二番目の節の一番目の項}
二番目の節の一番目の項の本文。
\subsection{二番目の節の二番目の項}
二番目の節の二番目の項の本文。
\end{document}
```

1.6.4 付録

文書の末尾には、必要に応じて、付録 (appendix) が置かれる場合があります。

付録という論理構造は、その先頭に、`\appendix` というコマンドを書くことによって記述されます。

文書クラスが `jarticle` の場合、節の番号は算用数字 (1, 2, 3, ···) で出力されますが、付録の節の番号は、大文字のアルファベット (A, B, C, ···) で出力されます。文書クラスが `jreport` や `jbook` の場合は、付録の章の番号がアルファベットになります。

LaTeX ソースの例 `appendix.tex`

```
\documentclass{jarticle}
\begin{document}
\section{本体の一番目の節}
\section{本体の二番目の節}
\appendix
\section{付録の一番目の節}
\section{付録の二番目の節}
\end{document}
```

1.6.5 引用文

文章の中に何らかの文章を引用する場合、その引用文が短いものならば、それを括弧で囲むだけで十分ですが、長い引用文の場合は、括弧で囲んだだけでは、どこからどこまでが引用文なのかということが視覚的に分かりにくくなります。

そこで、長い文章を引用する場合は、引用文の前後で改行して、引用文全体を少し右へ寄せることによって、どこからどこまでが引用文なのかということを視覚的に示すのが普通です。

文章の一部分が特殊な性格を持つことを視覚的に明示するような組版の方法は、印刷用語で「ディスプレイ」(display) と呼ばれます。LaTeX には、その内容がディスプレイの対象とみなされる環境がいくつかあります。

引用文をディスプレイの対象にしたいときは、quote という環境を使います。

LaTeX ソースの例 quote.tex

```
\documentclass{jarticle}
\begin{document}
ここはまだ地の文。
\begin{quote}
これは引用文。
\end{quote}
ここからは再び地の文。
\end{document}
```

1.6.6 箇条書き

ディスプレイの対象となる文章としては、箇条書き (list) と呼ばれるものもあります。

箇条書きというのは、記号や番号や見出しが先頭に付く文章の単位が一列に並んでできている文章のことです。箇条書きを構成するそれぞれの文章の単位は、「項目」(item) と呼ばれます。

LaTeX には、箇条書きを記述するための環境として、次の三つが定義されています。

itemize	• などの記号が項目の先頭に付く箇条書き。
enumerate	番号が項目の先頭に付く箇条書き。
description	見出しが項目の先頭に付く箇条書き。

箇条書きを構成するそれぞれの項目は、その先頭に\item というコマンドを書くことによって示されます。

\item は、引数を必要としないコマンドです。ただし、オプション引数を書くことは可能です。itemize 環境と enumerate 環境の場合、このコマンドのオプション引数は、記号や番号の代わりに項目の先頭に出力されます。description 環境の場合は、オプション引数が項目の見出しとして出力されます。

LaTeX ソースの例 list.tex

```
\documentclass{jarticle}
\begin{document}
三権分立の三権というのは、
\begin{itemize}
\item 立法権
\item 行政権
\item 司法権
\end{itemize}
の三つのことです。ジャイアンの本名は、
\begin{enumerate}
\item 源静香
\item 骨川スネ夫
\item 剛田武
\end{enumerate}
のうちのどれでしょう。次の飲み会は、
\begin{description}
\item[日時] 3 月 17 日午後 6 時
\item[場所] 居酒屋「ほげほげ」
\item[会費] 5,000 円
\end{description}
という予定です。
\end{document}
```

1.6.7 脚注

ページの下のほうに置かれる注釈は、「脚注」(footnote) と呼ばれます。

このページの下のほうに、脚注の実例[1]を作ってみました。このように、脚注は、本文の特定の部分に対する注釈になっていて、注釈の対象と注釈そのものとの対応関係は、番号によって示されます。

文章の一部分に対して脚注を付けたいときは、その部分の直後に\footnote というコマンドを書きます。このコマンドには、脚注となる文章を引数として記述します。

[1] これがそうです。

LaTeX ソースの例　footnote.tex

```
\documentclass{jarticle}
\begin{document}
私は、とある有名人\footnote{さて誰でしょう。}に似ています。
\end{document}
```

1.6.8　参考文献

文書の末尾には、しばしば参考文献 (reference) を列挙したもの、つまり書誌 (bibliography) を乗せることが必要になります。

書誌は、thebibliography という環境を使うことによって記述することができます。この環境は、箇条書きの形で書誌を出力するものだと考えることができます。それぞれの項目の先頭には、enumerate 環境と同じように番号が付きます。

それぞれの参照文献の項目は、普通の箇条書きの場合と同じように、その先頭に\item コマンドを書くことによって示すことができます。

thebibliography 環境を開始する\begin コマンドには、2 個の引数を書く必要があります。1 個目は環境の名前で、2 個目は数字の列です。この数字の列は、参考文献の番号が最大で何桁になるかということを環境に知らせるためのものです。通常、1 桁ならば 9、2 桁ならば 99、3 桁ならば 999、というように書きます。たとえば、

```
\begin{thebibliography}{99}
```

という\begin コマンドを書くことによって、参考文献の番号が最大で 2 桁になるということを LaTeX に知らせることができます。

LaTeX ソースの例　biblio.tex

```
\documentclass{jarticle}
\begin{document}
\begin{thebibliography}{9}
\item 田中花子『気象庁の大予言』。
\item 山田太郎『人間はつらいよ』。
\end{thebibliography}
\end{document}
```

\begin コマンドの 1 個目の引数は、環境を開始するために必要となるものなのに対して、2 個目以降の引数は、開始される環境が必要とするものです。ですから、\begin コマンドが何個の引数を必要とするかというのは、それによって開始される環境によって決まります。つまり、n 個の引数を必要とする環境を開始するためには、\begin コマンドに $n+1$ 個の引数を書く必要があるということです。

第 2 章　数式

2.1　数式の基礎

2.1.1　段落モードと数式モード

LaTeX は、通常は「段落モード」(paragraph mode) と呼ばれる方式で LaTeX ソースを処理するのですが、その中に含まれている数式 (formula) は、「数式モード」(math mode) と呼ばれる別の方式で処理します。

ですから、数式を出力するためには、LaTeX を数式モードに切り替えるための環境の中に数式の記述を書く必要があります。

2.1.2　インテキスト数式

LaTeX を数式モードに切り替える環境としては、いくつかのものがあるのですが、それらのうちでもっとも基本的なのは、math という環境です。

math は、「インテキスト数式」(in-text formula) と呼ばれる数式を出力する環境です。インテキスト数式というのは、地の文の中に組み込まれた数式のことです。

math 環境は、これまでに紹介した環境と同じように、\begin コマンドと\end コマンドを使って書くこともできますが、省略形を使って書くこともできます。math 環境の省略形は、$...$と

書きます。つまり、ドルマークで数式モードが始まって、次のドルマークで数式モードが終わるということです。

math 環境の中に、数字、英字、プラス (+)、マイナス (−)、イコール (=)、大なり (>)、小なり (<)、コンマ (,)、丸括弧 (( )) などから構成される数式を書くと、それは、そのまま数式として出力されます。たとえば、$a=b+c$と書けば、$a = b + c$ と出力されます。

また、数式モードでは、文字と文字のあいだの空白は無視されます。ですから、$x=y$と書いても、$x = y$と書いても、出力はどちらも $x = y$ です。

LaTeX ソースの例 math.tex

```
\documentclass{jarticle}
\begin{document}
地の文の中に数式を$x=a-(b+c)$というように出力したいときは、
その数式をドルマークで囲みます。
\end{document}
```

2.1.3 ディスプレイ数式

LaTeX を数式モードに切り替える環境としては、displaymath という環境もあります。

displaymath は、「ディスプレイ数式」(displayed formula) と呼ばれる数式を出力する環境です。ディスプレイ数式というのは、ディスプレイの対象にされた数式のことです。

displaymath 環境も、省略形を使って書くことができます。displaymath 環境の省略形は、\[...\] と書きます。つまり、\[で数式モードが始まって、\] で数式モードが終わるということです。たとえば、

```
\[ a=b+c \]
```

と書くことによって、

$$a = b + c$$

と出力することができます。

ただし、ディスプレイ数式は、通常はセンタリングされた位置に出力されます。先ほどの例のように、左端から少し右へ寄った位置にディスプレイ数式を出力するためには、文書クラスを指定するコマンドの中に、

```
\documentclass[fleqn]{jarticle}
```

というように、[fleqn] というオプション引数を書いておく必要があります。

LaTeX ソースの例 dismath.tex

```
\documentclass[fleqn]{jarticle}
\begin{document}
数式をディスプレイの対象にして、
\[ x=a-(b+c) \]
というように出力したいときは、
その数式を\verb/\[/と\verb/\]/で囲みます。
\end{document}
```

2.1.4 添字と肩字

数式では、しばしば添字や肩字が使われます。「添字」(subscript) というのは文字の右下に付加された小さな文字のことで（たとえば x_2 の 2)、「肩字」(superscript) というのは文字の右上に付加された小さな文字のことです（たとえば x^2 の 2)。

添字は、_（アンダースコア）というコマンドを使うことによって出力することができます。たとえば、x_{2}と書くことによって、x_{2} と出力することができます。

肩字は、^（サーカムフレックス）というコマンドを使うことによって出力することができます。たとえば、x^{2}と書くことによって、x^{2} と出力することができます。

ひとつの数式に添字と肩字の両方を付加する場合、x_{2}^{2}と書くと、x_{2}^{2} というように添字と肩字が上下に並んでしまって、数式の構造が分かりにくくなります。このような場合は、中括弧を使うことによって数式の構造を明示するといいでしょう。たとえば、{x_{2}}^{2}と書けば ${x_{2}}^{2}$ と出力されて、{x^{2}}_{2}と書けば ${x^{2}}_{2}$ と出力されます。

LaTeX ソースの例 subsuper.tex

```
\documentclass{jarticle}
\begin{document}
\[ A_{B}, A^{B} \]
\[ A_{B_{C}}, A^{B^{C}} \]
\[ A_{B}^{C}, {A_{B}}^{C}, {A^{C}}_{B} \]
\end{document}
```

2.1.5 分数

インテキスト数式の中に分数 (fraction) を書く場合は、通常、/（スラッシュ）という文字を使います。たとえば、`$1/a$`と書くことによって、$1/a$ と出力することができます。

ディスプレイ数式の中に分数を書く場合は、`\frac` というコマンドを使います。このコマンドには、二つの引数が必要です。1 個目の引数が分子 (numerator) になって、2 個目の引数が分母 (denominator) になります。たとえば、

```
\[ \frac{c+d}{a+b} \]
```

と書くことによって、

$$\frac{c+d}{a+b}$$

と出力することができます。

LaTeX ソースの例 `frac.tex`

```
\documentclass{jarticle}
\begin{document}
\[ F = G\frac{m_{1}m_{2}}{r^{2}} \]
\end{document}
```

2.1.6 平方根

平方根 (square root) は、`\sqrt` というコマンドを使うことによって出力することができます。たとえば、`$\sqrt{x}$`と書くことによって、$\sqrt{x}$ と出力することができます。

`\sqrt` コマンドにオプション引数を書くことによって、べき乗根を出力することもできます。たとえば、`$\sqrt[n]{x}$`と書くことによって、$\sqrt[n]{x}$ と出力することができます。

LaTeX ソースの例 `sqrt.tex`

```
\documentclass{jarticle}
\begin{document}
\[ S = \sqrt{s(s-a)(s-b)(s-c)} \]
\[ \sqrt[n]{a} = a^{\frac{1}{n}} \]
\end{document}
```

2.2 数学記号

2.2.1 この節について

数式モードでは、さまざまな数学記号 (mathematical symbol) を出力するコマンドを使うことができます。この節では、数学記号を出力するコマンドを紹介したいと思います。

ただし、数学記号を出力するコマンドは無数にあって、ここで紹介するコマンドは、それらのうちのほんの一部です。ここで紹介していないコマンドについては、市販されている書籍などを参照してください。

2.2.2 二項演算子

二項演算子 (binary operator) を出力するコマンドとしては、次のようなものがあります。

`\pm`	$\pm$	`\mp`	$\mp$	`\times`	$\times$	`\div`	$\div$	`\circ`	$\circ$
`\bullet`	$\bullet$	`\wedge`	$\wedge$	`\vee`	$\vee$	`\cap`	$\cap$	`\cup`	$\cup$

2.2.3 関係演算子

関係演算子 (relational operator) を出力するコマンドとしては、次のようなものがあります。

\neq	$\neq$	\ge	$\geq$	\le	$\leq$	\equiv	$\equiv$	\supset	$\supset$
\subset	$\subset$	\supseteq	$\supseteq$	\subseteq	$\subseteq$	\in	$\in$	\ni	$\ni$

LaTeX ソースの例 operator.tex

```
\documentclass{jarticle}
\begin{document}
$ A \times B, A \div B, A \cap B, A \cup B, A \wedge B,
A \vee B, A \ge B, A \le B, A \equiv B, A \supset B,
A \subset B, a \in A , A \ni a $
\end{document}
```

2.2.4 斜線

関係が成り立っていないという意味の斜線 (slash) を関係演算子の上に重ねて出力したいときは、\not というコマンドを使います。このコマンドの直後に関係演算子を出力するコマンドを書くことによって、その関係演算子の上に斜線を重ねて出力することができます。

LaTeX ソースの例 not.tex

```
\documentclass{jarticle}
\begin{document}
\[ A \not\equiv B, A \not\subset B, x \not\in B \]
\end{document}
```

2.2.5 否定演算子など

否定演算子 (not operator)、全称記号 (universal quantifier)、存在記号 (existential quantifier)、無限大 (infinity)、アレフ (aleph) は、次のコマンドを使うことによって出力することができます。

\neg	$\neg$	\forall	$\forall$	\exists	$\exists$	\infty	∞	\aleph	$\aleph$

2.2.6 プライム

数式モードでアポストロフィーを書くと、それはプライム[1](prime) として出力されます。n 個のアポストロフィーを連続して書けば、n 重のプライムになります。たとえば、a' と書けば a'、a'' と書けば a''、a''' と書けば a''' と出力されます。

2.2.7 矢印

矢印 (arrow) を出力するコマンドとしては、次のようなものがあります。

\leftarrow	$\leftarrow$	\rightarrow	$\rightarrow$	\leftrightarrow	$\leftrightarrow$
\longleftarrow	$\longleftarrow$	\longrightarrow	$\longrightarrow$	\longleftrightarrow	$\longleftrightarrow$
\Leftarrow	$\Leftarrow$	\Rightarrow	$\Rightarrow$	\Leftrightarrow	$\Leftrightarrow$

2.2.8 省略記号

省略記号 (ellipsis) を出力するコマンドとしては、次のようなものがあります。

\ldots	$\ldots$	\cdots	$\cdots$	\vdots	$\vdots$	\ddots	$\ddots$

\ldots コマンドは、$x_1, x_2, \ldots, x_n$ というように列挙を省略する場合などに使われる省略記号で、\cdots コマンドは、$x_1 + x_2 + \cdots + x_n$ というように演算を省略する場合などに使われる省略記号です。

\ldots コマンドは、数式モード以外のモードでも使うことができます。

2.2.9 ギリシア文字

ギリシア文字 (Greek letter) を出力するコマンドとしては、次のようなものがあります。

\gamma	γ	\delta	δ	\theta	θ	\lambda	λ	\xi	ξ
\Gamma	Γ	\Delta	Δ	\Theta	Θ	\Lambda	Λ	\Xi	Ξ

[1] 通常、日本語ではプライムは「ダッシュ」と呼ばれます。

同じ形のラテン文字がある、小文字の o と、大文字の A、B、E、Z、H、I、K、M、N、O、P、T、Y、X については、コマンドはありません。

2.2.10 log 型関数

log、lim、sin、cos などの関数は、総称して「log 型関数」(log-like function) と呼ばれます。log 型関数を出力するコマンドとしては、次のようなものがあります。

`\log` log　`\lim` lim　`\sin` sin　`\cos` cos　`\det` det

ディスプレイ数式では、`\lim` コマンドの直後に添字のコマンドを書くことによって、$x \to \infty$ のような式を lim の真下に出力することができます。

LaTeX ソースの例　`log.tex`

```
\documentclass{jarticle}
\begin{document}
\[ \lim_{x\rightarrow\infty}f(x) \]
\end{document}
```

2.2.11 総和など

総和 (summation) などを出力するコマンドとしては、次のようなものがあります。

`\sum` $\sum$　`\bigcap` $\bigcap$　`\bigcup` $\bigcup$　`\int` $\int$

ディスプレイ数式では、`\sum` コマンドの直後に書かれた添字と肩字のコマンドは、$\sum$ の真下と真上に式を出力します。

LaTeX ソースの例　`sum.tex`

```
\documentclass{jarticle}
\begin{document}
\[ \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \]
\end{document}
```

2.2.12 区切り記号

数式の範囲を示すために、その左端と右端に置かれる記号は、区切り記号 (delimiter) と呼ばれます。

丸括弧 (( )) と角括弧 ([ ]) と縦棒 (|) は、それらの文字を書けばその文字がそのまま出力されます。中括弧 ({ }) は、`\{`と`\}`というコマンドを書くことによって出力することができます。

高さの高い数式を区切り記号で囲みたいときは、`\left` と`\right` というコマンドを使います。`\left` コマンドの直後に (、[、|、`\{`などを書くと左側の区切り記号が出力されて、`\right` コマンドの直後に)、]、|、`\}`などを書くと右側の区切り記号が出力されます。`\left` と`\right` は、囲まれる数式の高さに合わせて、出力する区切り文字の高さを調整します。

`\left` と`\right` は、かならずペアで使う必要があります。左側だけ、あるいは右側だけに区切り記号を出力したい場合は、出力したくないほうのコマンドの直後にドット (.) を書きます。

LaTeX ソースの例　`delim.tex`

```
\documentclass{jarticle}
\begin{document}
\[ ( \frac{B}{A} ) \]
\[ \left( \frac{B}{A} \right) \]
\[ \left\{ \frac{B}{A} \right. \]
\[ \left. \frac{B}{A} \right\} \]
\end{document}
```

2.3 配列

2.3.1 配列とは何か

LaTeX では、縦方向と横方向に数式を並べたもののことを「配列」(array) と呼びます。

配列は、複数の数式を整然と並べたい場合、行列 (matrix) や行列式 (determinant) を出力したい場合などに使われます。

配列の一部分であって、数式が横方向に一列に並んでいるもののことを、その配列の「行」(column) と呼びます。それに対して、数式が縦方向に一列に並んでいるもののことを、その配列の「列」(row) と呼びます。行は、上から順番に 1 行目、2 行目、3 行目、・・・と呼ばれ、列は、左から順番に 1 列目、2 列目、3 列目、・・・と呼ばれます。

2.3.2　array 環境

配列は、array という環境を使うことによって出力することができます。

array 環境は、1 個の引数を必要とします（つまり、array 環境を開始する\begin コマンドには 2 個の引数を書く必要があるということです）。array 環境が必要とする引数というのは、配列を構成するそれぞれの列について、それを左に寄せるか右に寄せるかセンタリングするかということを指定する文字列です。

左寄せ、右寄せ、センタリングを指定する文字列は、l、r、c という文字を並べることによって作ります。l が左寄せ、r が右寄せ、c がセンタリングです。たとえば、lrc という文字列は、1 列目を左寄せ、2 列目を右寄せ、3 列目をセンタリングするという意味になります。

配列の一つの行を構成するそれぞれの数式は、&（アンパサンド）で区切ります。そして、配列を構成するそれぞれの行は、\\コマンドで区切ります。

LaTeX ソースの例　array.tex

```
\documentclass{jarticle}
\begin{document}
\[
\begin{array}{lrc}
a     & z     & i     \\
a+b   & y+z   & i+j   \\
a+b+c & x+y+z & i+j+k
\end{array}
\]
\end{document}
```

2.3.3　行列と行列式

行列 (matrix) と行列式 (determinant) を囲む括弧は、第 2.2 節で紹介した\left と\right というコマンドを使うことによって出力することができます。

LaTeX ソースの例　matrix.tex

```
\documentclass{jarticle}
\begin{document}
\[
A = \left(
\begin{array}{ccc}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{array}
\right)
\]
\[
\det A = \left|
\begin{array}{ccc}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{array}
\right|
\]
\end{document}
```

2.3.4　場合分け

場合分け (cases) も、array 環境を使うことによって出力することができます。

LaTeX ソースの例　cases.tex

```
\documentclass{jarticle}
```

```
\begin{document}
\[
F_{n} = \left\{
\begin{array}{ll}
1                  & (n=0)      \\
1                  & (n=1)      \\
F_{n-2} + F_{n-1} & (n \ge 2)
\end{array}
\right.
\]
\end{document}
```

2.4 数学記号の積み重ね

2.4.1 上線

この節では、数学記号を上下に積み重ねて出力するためのコマンドをいくつか紹介したいと思います。

まず最初は上線 (overline) と下線 (underline) です。

上線は\overline、下線は\underline というコマンドを使うことによって出力することができます。これらのコマンドは、引数で出力された数式の上に水平な線を引きます。

\underline コマンドは、数式モードだけではなくて、それ以外のモードでも使うことができます。

LaTeX ソースの例 overline.tex

```
\documentclass{jarticle}
\begin{document}
\[ \overline{P \cup \overline{Q}} \cap R \]
We can draw \underline{underline} anywhere.
\end{document}
```

2.4.2 上下の中括弧

\overbrace と\underbrace というコマンドは、引数で出力された数式の上下に中括弧を出力します。

\overbrace コマンドの直後に肩字を書くと、その肩字は中括弧の真上に出力されます。同じように、\underbrace コマンドの直後に添字を書くと、その添字は中括弧の真下に出力されます。

LaTeX ソースの例 brace.tex

```
\documentclass{jarticle}
\begin{document}
\[ \overbrace{A+\underbrace{B+C+D+E}+F+G} \]
\[ \underbrace{H+I+\overbrace{J+K+L+M}^{x}+N}_{y} \]
\end{document}
```

2.4.3 文字の上に乗せる記号

次のようなコマンドを使うことによって、文字の上に記号を乗せることができます。

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

i または j の上に記号を乗せる場合は、\imath と\jmath というコマンドを使うことによって、$\imath$ と $\jmath$ という、本来の形から点を取り除いた文字を出力する必要があります。

LaTeX ソースの例 vector.tex

```
\documentclass{jarticle}
\begin{document}
$ \vec{a}, \acute{a}, \grave{a}, \bar{a}, \dot{a},
\ddot{a}, \breve{a}, \tilde{a}, \hat{a}, \check{a},
\vec{\imath}, \bar{\jmath} $
\end{document}
```

第3章 表

3.1 表の基礎

3.1.1 表とは何か

LaTeX では、縦方向と横方向に記述を並べたもののことを「表」(table) と呼びます。

表の中に並べられたそれぞれの記述は、「項目」(item) と呼ばれます。

表の一部分であって、項目が横方向に一列に並んでいるもののことを、その表の「行」(column) と呼びます。それに対して、項目が縦方向に一列に並んでいるもののことを、その表の「列」(row) と呼びます。行は、上から順番に 1 行目、2 行目、3 行目、・・・と呼ばれ、列は、左から順番に 1 列目、2 列目、3 列目、・・・と呼ばれます。

3.1.2 tabular 環境

表は、tabular という環境を使うことによって出力することができます。

tabular 環境というのは、第 2.3 節で紹介した array 環境とほとんど同じ機能を持つ環境です。それらのあいだの最大の相違点は、array 環境は数式モードだけでしか使うことができないのに対して、tabular 環境は数式モード以外のモードでも使うことができるというところにあります。

array 環境と同じように、tabular 環境も、表を構成するそれぞれの列について、それを左に寄せるか右に寄せるかセンタリングするかということを指定する文字列を、引数として書く必要があります。その文字列の作り方も array 環境と同じで、l（左寄せ）、r（右寄せ）、c（センタリング）という文字を並べることによって作ります。

項目と行を区切る方法も array 環境と同じです。項目は &（アンパサンド）で区切り、行は\\コマンドで区切ります。

LaTeX ソースの例 tabular.tex

```
\documentclass{jarticle}
\begin{document}
\begin{tabular}{lrc}
空      & 山      & 海      \\
太陽    & 人間    & 野菜    \\
自動車 & 蛍光灯 & 映画館
\end{tabular}
\end{document}
```

3.1.3 表の垂直方向の位置

LaTeX は、表を一つの文字とみなして配置します。文字としての表は、行の中心線と表の中心線が一致する位置に出力されます。つまり、$\begin{array}{c}1\text{列目}\\2\text{列目}\end{array}$ という位置に置かれるわけです。

表の垂直方向の位置は、tabular 環境のオプション引数を書くことによって指定することができます。角括弧の中に、

t 表の 1 行目と表の前後の行とを一致させる。

b 表の最後の行と表の前後の行とを一致させる。

のどちらかの文字を書けば、その文字で指定された位置に表が出力されます。

コマンドのオプション引数はコマンド名の直後に書くわけですが、環境のオプション引数は、\begin コマンドの 1 個目の引数の直後に書きます。ですから、表の 1 行目と表の前後の行とが一致する位置に表を出力したいときは、

```
\begin{tabular}[t]{lrc}
```

というような\begin コマンドを書くことになります。

LaTeX ソースの例 valign.tex

```
\documentclass{jarticle}
\begin{document}
```

```
表は、デフォルトだと
\begin{tabular}{l}1 行目\\2 行目\end{tabular}という位置に
出力されますが、
\begin{tabular}[t]{l}1 行目\\2 行目\end{tabular}という位置や、
\begin{tabular}[b]{l}1 行目\\2 行目\end{tabular}という位置に
出力することも可能です。
\end{document}
```

3.2 罫線

3.2.1 垂直の罫線

この節では、表の中に罫線 (ruled line) を引く方法について説明したいと思います。

垂直の罫線は、tabular 環境の引数の中に縦棒 (|) を書くことによって引くことができます。たとえば、{cc|c|}という引数を書けば、

1 列目	2 列目	3 列目

というように、3 列目の左側と右側に垂直の罫線が引かれます。

|| というように、2 個の縦棒を連続して書くことによって、垂直の罫線を二重に引くことができます。たとえば、{c||cc}という引数を書けば、

1 列目	2 列目	3 列目

というように、1 列目と 2 列目のあいだに、二重になった垂直の罫線が引かれます。

LaTeX ソースの例　vline.tex

```
\documentclass{jarticle}
\begin{document}
\begin{tabular}{|c||c|r|}
     & 性別 & 年齢 \\
太郎 &  男  &   22 \\
花子 &  女  &   23
\end{tabular}
\end{document}
```

3.2.2 水平の罫線

水平の罫線は、\hline というコマンドを書くことによって引くことができます。たとえば、

```
\begin{tabular}{c}
1 行目 \\ \hline
2 行目
\end{tabular}
```

というように、1 行目と 2 行目のあいだの改行の直後に\hline コマンドを書くと、

1 行目
2 行目

というように、1 行目と 2 行目とのあいだに水平の罫線が引かれます。

1 行目の上に水平の罫線を引きたいときは、1 行目よりも前に\hline コマンドを書きます。また、最後の行の下に水平の罫線を引きたいときは、最後の行の末尾に\\コマンドを書いて、その後ろに\hline コマンドを書きます。たとえば、

```
\begin{tabular}{c}\hline
1 行目 \\
2 行目 \\ \hline
\end{tabular}
```

という tabular 環境を書くと、

1 行目
2 行目

というように、1 行目の上と 2 行目の下に水平の罫線が引かれます。

2 個の\hline を連続して書くことによって、水平の罫線を二重に引くことができます。たとえば、

```
\begin{tabular}{c}
1 行目 \\ \hline\hline
```

```
    2行目
    \end{tabular}
```

という tabular 環境を書くと、$\begin{array}{c}\text{1行目}\\ \hline\hline \text{2行目}\end{array}$ というように、1行目と2行目のあいだに、二重になった水平の罫線が引かれます。

LaTeX ソースの例　hline.tex

```
\documentclass{jarticle}
\begin{document}
\begin{tabular}{ccr}  \hline
     & 性別 & 年齢 \\ \hline\hline
太郎 &  男  &   22 \\ \hline
花子 &  女  &   23 \\ \hline
\end{tabular}
\end{document}
```

3.2.3 水平の部分的な罫線

水平の罫線を引くコマンドは、\hline だけではなくて、もうひとつあります。それは、\cline というコマンドです。

\hline が表の左端から右端までの罫線を引くのに対して、\cline が罫線を引くのは指定された範囲だけです。その範囲は、{i-j}という引数で指定します。i は罫線を開始する列の番号で、j は罫線を終了する列の番号です。たとえば、\cline{3-5}というコマンドを書くと、3列目から5列目までの水平の罫線が引かれます。

LaTeX ソースの例　cline.tex

```
\documentclass{jarticle}
\begin{document}
\begin{tabular}{cccccc}  \cline{2-5}
1 & 2 & 3 & 4 & 5 & 6 \\ \cline{1-2}\cline{5-6}
1 & 2 & 3 & 4 & 5 & 6 \\ \cline{1-1}\cline{3-4}\cline{6-6}
\end{tabular}
\end{document}
```

3.3 表に関するエトセトラ

3.3.1 複数の列にまたがる項目

表を作るとき、しばしば、複数の列にまたがる項目を作ることが必要になる場合があります。つまり、

1列目	2列目	3列目
3列にまたがる項目		

というような表を作ることが必要になる場合です。

複数の列にまたがる項目は、\multicolumn というコマンドを書くことによって作ることができます。このコマンドには、3個の引数を書く必要があります。1個目はまたがる列の数、2個目は水平方向の位置などを示す文字列（l、r、c、| で作ります）、そして3個目は項目そのものです。たとえば、

```
\multicolumn{4}{c}{ほげほげ}
```

というコマンドを書くことによって、4列にまたがっていてセンタリングされた「ほげほげ」という項目を作ることができます。

\multicolumn コマンドは、複数の列にまたがる項目を作りたい場合だけではなくて、特定の項目だけに対して水平方向の位置を指定したい場合にも使うことができます。

LaTeX ソースの例　multicol.tex

```
\documentclass{jarticle}
\begin{document}
\begin{tabular}{|c|c|c|}\hline
項目 & センタリング & 項目 \\ \hline
```

```
\multicolumn{2}{|c|}{2 列にまたがる項目} & 項目 \\ \hline
項目 & \multicolumn{2}{c|}{2 列にまたがる項目} \\ \hline
項目 & \multicolumn{1}{r|}{右寄せ} & 項目 \\ \hline
\end{tabular}
\end{document}
```

3.3.2 表の中の表

表の中の項目として tabular 環境を書くことによって、複数の行から構成される項目を作ることができます。

LaTeX ソースの例　tabintab.tex

```
\documentclass{jarticle}
\begin{document}
\begin{tabular}{|c|c|}\hline
項目 &
\begin{tabular}{c}項目の 1 行目\\項目の 2 行目\end{tabular}
\\ \hline
\end{tabular}
\end{document}
```

3.3.3 列の横幅の設定

列の横幅は、項目の長さに基づいて自動的に決定されるのですが、特定の長さを設定することも可能です。横幅が設定された列の項目が、その横幅よりも長い場合、入り切らなかった部分は次の行へ送られます。

列の横幅を設定したいときは、tabular 環境の引数の中に、l や r や c の代わりとして、

```
p{長さ}
```

という形の記述を書きます。そうすると、その記述の中に書かれた長さが、列の横幅として設定されることになります。

LaTeX での長さの記述は、10 進数と単位とを組み合わせたものです。長さの単位としては、次のようなものを使うことができます。

- mm　ミリメートル。
- cm　センチメートル。
- in　インチ（1 インチは 2.54 センチメートル）。
- pt　ポイント（1 ポイントは 1/72.27 インチ、0.3514 ミリメートル）。
- em　“M” という文字の横幅。
- ex　“x” という文字の高さ。
- zw　日本語の文字の横幅。

たとえば、tabular 環境を開始する\begin コマンドを、

```
\begin{tabular}{ccp{54mm}cc}
```

と書くことによって、3 列目に対して 54 ミリメートルという横幅を設定することができます。

横幅が設定された列は、デフォルトでは左寄せになっています。右に寄せたりセンタリングしたりするためには、\multicolumn コマンドを使う必要があります。

LaTeX ソースの例　width.tex

```
\documentclass{jarticle}
\begin{document}
\begin{tabular}{|c|p{3cm}|}\hline
項目 & わかっちゃいるけどやめられない。 \\ \hline
項目 & \multicolumn{1}{r|}{右寄せ} \\ \hline
項目 & \multicolumn{1}{c|}{センタリング} \\ \hline
\end{tabular}
\end{document}
```

第 4 章　相互参照と目次と索引

4.1　相互参照

4.1.1　相互参照とは何か

文章の中には、しばしば、「第何章を参照してください」というような、同じ文章の別の部分を参照するようにと読者に指示する記述が含まれています。このような記述を書くためには、参照先の番号が必要になるわけですが、その番号は、文章を書き進める過程や文章の改訂などで変化する可能性があります。

文章の部分の番号と、それを参照する記述の中の番号とを一致させるという作業は、「相互参照」(cross-reference) と呼ばれます。

相互参照は、文章が長くなればなるほど面倒なものににになっていきます。しかし、LaTeX には相互参照を自動的に実行する機能がありますので、それを使うことによって、この面倒な作業を LaTeX に任せてしまうことができます。

4.1.2　ラベル

相互参照を LaTeX に実行させるためには、参照される部分に対して、「ラベル」(label) と呼ばれるものを与える必要があります。ラベルというのは、文章の部分を識別するための名前のことです。

文章の部分に対してラベルを与えたいときは、`\label` というコマンドを書きます。このコマンドには、文章の部分に与えるラベルを引数として書く必要があります。ラベルに使うことのできる文字は、ほとんどすべてと言っていいのですが、`\` と `{` と `}` だけは使うことができません。

`\label` コマンドを書く場所は、ラベルを与える部分の中です。セクション単位に対してラベルを与えたいときは、通常、そのセクション単位のセクショニングコマンドの直後に`\label` コマンドを書きます。たとえば、

```
\section{彼女の好きな虫}\label{sec:insect}
```

と書くことによって、「彼女の好きな虫」という節に対して `sec:insect` というラベルを与えることができます。

4.1.3　番号の生成

文章の部分に対してラベルが与えられていれば、LaTeX は、その部分の番号をどこででも生成することができます。

ラベルが与えられている部分の番号を生成したいときは、`\ref` というコマンドを書きます。このコマンドの引数として、参照したい部分に与えられているラベルを書けば、その部分の番号が生成されます。たとえば、もしも、 `sec:insect` というラベルが与えられている部分の番号が 8 だとするとき、

```
第\ref{sec:insect}節を参照してください。
```

と書いたとすると、LaTeX は、

```
第 8 節を参照してください。
```

というように、その番号を生成します。

LaTeX ソースの例　`cross.tex`

```
\documentclass{jarticle}
\begin{document}
\section{桃太郎}\label{sec:momo}
一寸法師については第\ref{sec:issun}節を参照してください。
\section{一寸法師}\label{sec:issun}
桃太郎については第\ref{sec:momo}節を参照してください。
\end{document}
```

この LaTeX ソースをコンパイルすると、

```
LaTeX Warning: There were undefined references.
```

というような、いくつかの警告 (warning) が出力されます。実は、LaTeX に相互参照を実行させるためには、2 回のコンパイルが必要なのです。1 回目でラベルと番号をファイルに書き込んで、2 回目で番号を生成します。

4.2 目次

4.2.1 目次の作り方

LaTeX は、セクショニングコマンドから自動的に目次 (table of contents) を作成するという機能を持っています。

LaTeX に目次を作らせたいときは、`\tableofcontents` というコマンドを、目次を出力したい場所に書きます。

LaTeX ソースの例　`contents.tex`

```
\documentclass{jarticle}
\begin{document}
\tableofcontents
\section{一番目の節}
\subsection{一番目の節の一番目の項}
\subsection{一番目の節の二番目の項}
\section{二番目の節}
\subsection{二番目の節の一番目の項}
\subsection{二番目の節の二番目の項}
\end{document}
```

相互参照の場合と同じように、LaTeX に目次を作成させる場合も、2 回のコンパイルが必要です。1 回目で目次を作成するために必要な情報をファイルに書き込んで、2 回目で目次を作成します。

4.2.2 目次の深さ

どれくらい小さなセクション単位まで目次を出力するのかということを、目次の「深さ」(depth) と呼びます。

目次の深さを設定したいときは、

```
\setcounter{tocdepth}{レベル番号}
```

というコマンドをプリアンブルに書きます（「プリアンブル」(preamble) というのは、第 1.3 節で説明したように、`document` 環境よりも上の部分のことです)。

目次の深さを設定するコマンドの中に書く「レベル番号」(level number) というのは、セクション単位の大きさを示す番号のことです。レベル番号は、文書クラスごとに違っているのですが、`jarticle` の場合は、節 (section) が 1 で、項 (subsection) が 2 です。

LaTeX ソースの例　`tocdepth.tex`

```
\documentclass{jarticle}
\setcounter{tocdepth}{1}
\begin{document}
\tableofcontents
\section{一番目の節}
\subsection{一番目の節の一番目の項}
\subsection{一番目の節の二番目の項}
\section{二番目の節}
\subsection{二番目の節の一番目の項}
\subsection{二番目の節の二番目の項}
\end{document}
```

4.3 索引

4.3.1 mendex

この節では、LaTeX を使って索引 (index) を作成する方法について説明したいと思います。

ところで、すでに説明したように、相互参照や目次は、LaTeX に作ってもらうことができます。しかし、索引は、LaTeX だけでは作ることができません。索引を作るためには、LaTeX だけではなくて、mendex というソフトウェアの力を借りる必要があります。

ちなみに、mendex は、MakeIndex というソフトウェアを改造して、日本語が使えるようにしたものです。日本語の文字を含まない索引を作る場合は、mendex を使ってもかまいませんし、MekeIndex を使ってもかまいません。

4.3.2　索引を作るためのコマンド

LaTeX と mendex を使って索引を作りたいときは、LaTeX ソースのプリアンブルの中に、

```
\usepackage{makeidx}
\makeindex
```

という二つのコマンドを書いておく必要があります。そしてさらに、document 環境の中に、

```
\printindex
```

というコマンドを書いておくと、そのコマンドの場所に索引が出力されます。

4.3.3　索引に言葉を載せるコマンド

索引に言葉を載せるためには、\index というコマンドを、その言葉が登場する場所に書いておく必要があります。

\index は、1 個の引数を必要とするコマンドです。引数として何らかの言葉を書くと、その言葉が索引に載ることになります。たとえば、

```
くいだおれ\index{くいだおれ}
```

というように、「くいだおれ」という言葉の直後に、その言葉を索引に載せるコマンドを書いたとしましょう。もしも、この「くいだおれ」が 24 ページ目に出力されたとすると、索引には、

くいだおれ, 24

という項目が掲載されることになります。

索引に載せたい言葉が、英数字、ひらがな、カタカナでできている場合は、その言葉だけを \index コマンドの引数として書けばいいのですが、漢字を含んでいる言葉を索引に載せるためには、その言葉に加えて、その読み方をひらがなで書いておく必要があります。言葉とその読み方は、

```
\index{かんじ@漢字}
```

というように、真ん中にアットマークを書いて、その右側に言葉、左側に読み方を書きます。

4.3.4　索引を作成する手順

索引を作成するためには、次のような三段階の作業をする必要があります。

(1) LaTeX ソースを LaTeX でコンパイルします。そうすると、索引を作るための情報が、「idx ファイル」(idx file) と呼ばれるファイルに書き込まれます。idx ファイルのファイル名は、LaTeX ソースのファイル名の拡張子を .idx に変更したものになります。

(2) idx ファイルを mendex で処理します。mendex を起動するコマンドは、idx ファイルの名前が namako.idx だとするならば、

```
mendex namako.idx
```

と書きます（拡張子の .idx は省略することもできます）。idx ファイルを mendex で処理すると、索引が作成されて、その LaTeX ソースが、「ind ファイル」(ind file) と呼ばれるファイルに書き込まれます。ind ファイルの名前は、idx ファイルのファイル名の拡張子を .ind に変更したものになります。

(3) もう一度、LaTeX ソースを LaTeX でコンパイルします。そうすると、ind ファイルが読み込まれて、\printindex コマンドの位置に索引が出力されます。

それでは、次の LaTeX ソースを入力して、上の手順で処理してみてください。

LaTeX ソースの例　index.tex

```
\documentclass{jarticle}
\usepackage{makeidx}
\makeindex
\begin{document}
コマンド\index{コマンド}、
くいだおれ\index{くいだおれ}、
Arcadia\index{Arcadia}、
関西人\index{かんさいじん@関西人}
\printindex
\end{document}
```

ちなみに、索引は 2 ページ目に出力されます。

第 5 章　コマンドと環境の定義

5.1　コマンドの定義

5.1.1　この章について

LaTeX は、新しいコマンド名や新しい環境名に意味を与えることができるという機能を持っています。また、すでに存在するコマンド名や環境名の意味を変更することも可能です。

新しいコマンド名または環境名に意味を与えることを、コマンドまたは環境の「定義」(definition) と呼びます。また、すでに意味が与えられているコマンド名または環境名の意味を変更することを、コマンドまたは環境の「再定義」(redefinition) と呼びます。

この章では、コマンドや環境を定義したり再定義したりする方法について説明したいと思います。

5.1.2　コマンド名の作り方

コマンド名は、1 個のバックスラッシュの後ろに 1 個以上の英字または日本語の文字を並べることによって作ります。たとえば、

```
\kangaroo    \DNA    \CogitoErgoSum    \冥王星
```

などは、正しいコマンド名です。

英字の大文字と小文字は区別されますので、\who と\WHO とは異なるコマンド名とみなされます。

コマンド名の中に数字や特殊文字を混ぜることはできませんので、

```
\Alien5    \carpe-diem    \carpe_diem
```

などをコマンド名として使うことはできません。ただし、バックスラッシュの後ろに数字または特殊文字を 1 個だけ書いたもの（たとえば\7 や\+など）をコマンド名として使うことは可能です。

5.1.3　コマンドを定義するコマンド

コマンドを定義したいときは、\newcommand というコマンドを使います。このコマンドは、document 環境の中に書くこともできますが、通常はプリアンブルに書きます。

\newcommand コマンドは、

```
\newcommand{コマンド名}{文字列}
```

というように、2 個の引数を必要とします。「コマンド名」のところには、意味を与えたいコマンド名を書きます。そして、「文字列」のところには、コマンド名の意味となる文字列を書きます。

\newcommand コマンドは、2 個目の引数に対して、その名前として 1 個目の引数を与えます。実は、コマンドを定義するというのは、ただ単に文字列に名前を与えるということなのです。

LaTeX というのは、LaTeX ソースを、TeX に処理させることのできる形に変換するソフトウェアです。LaTeX は、LaTeX ソースの中に書かれているコマンドを、そのコマンドの意味となっている文字列に置き換えます。コマンドを置き換えた結果がさらにコマンドを含んでいる場合は、それも意味に置き換えます。そのようにしてコマンドを置き換えていった結果が、TeX によって処理されるのです。

たとえば、

```
\newcommand{\recipient}{花子}
```

というコマンドは、「花子」という文字列に対して\recipient というコマンド名を与えます。ですから、このコマンドよりも後ろに、

```
私は\recipient さんを愛しています。
```

と書いたとすると、この文は LaTeX によって、

私は花子さんを愛しています。

に変換されて、その結果が TEX によって処理されます。

コマンドを含んでいる文字列に対してコマンド名を与えることも可能です。たとえば、

```
\newcommand{\recipientgt}{\textgt{花子}}
```

というコマンドで定義された\recipientgt というコマンドは、**花子**というように、「花子」という文字列をゴシック体で出力します。

\small や\large のような宣言（第 1.5 節参照）を含む文字列にコマンド名を与える場合は、注意が必要です。たとえば、花子というように、「花子」という文字列を大きな文字で出力するコマンドを定義しようとして、

```
\newcommand{\recipientlarge}{\large 花子}
```

というコマンドを書いたとしましょう。そののち、

```
私が好きなのは\recipientlarge さんだけです。
```

という文を書いたとすると、この文は、

```
私が好きなのは\large 花子さんだけです。
```

に変換されますので、「花子」だけではなくて、その後ろの部分まで文字が大きくなってしまいます。この場合は、

```
\newcommand{\recipientlarge}{{\large 花子}}
```

というように、宣言の影響が及ぶようにしたい範囲を中括弧で囲む必要があります。

LaTeX ソースの例　newcomm.tex

```
\documentclass{jarticle}
\newcommand{\recipientgt}{\textgt{花子}}
\newcommand{\recipientlarge}{{\large 花子}}
\begin{document}
私は\recipientgt さんを愛しています。
私が好きなのは\recipientlarge さんだけです。
\end{document}
```

5.1.4　コマンドを再定義するコマンド

コマンドを定義したいときは\newcommand コマンドを使えばいいわけですが、このコマンドを使ってコマンドを再定義することはできません。たとえば、

```
\newcommand{\small}{Small is beautiful.}
```

というように、すでに意味が与えられているコマンド名を 1 個目の引数として書いたとすると、LaTeX でコンパイルしたときにエラーが発生します。

コマンドを再定義したいときは、\newcommand コマンドではなくて、\renewcommand というコマンドを使う必要があります。

\renewcommand コマンドの使い方は、\newcommand コマンドとまったく同じです。1 個目の引数は、意味を変更したいコマンド名で、2 個目の引数は、新しい意味となる文字列です。

LaTeX ソースの例　renewco.tex

```
\documentclass{jarticle}
\renewcommand{\small}{Small is beautiful.}
\begin{document}
私たちの合言葉は\small です。
\end{document}
```

5.2　環境の定義

5.2.1　環境名の作り方

環境名は、1 個以上の英字または日本語の文字を並べることによって作ります。たとえば、

```
anteater    LCL    JactaAleaEst    銀河系
```

などは、正しい環境名です。

英字の大文字と小文字は区別されますので、ice と ICE とは異なる環境名とみなされます。

5.2.2 環境を定義するコマンド

環境を定義したいときは、\newenvironment というコマンドを使います。このコマンドは、document 環境の中に書くこともできますが、通常はプリアンブルに書きます。

\newenvironment コマンドは、

\newenvironment{環境名}{開始文字列}{終了文字列}

というように、3 個の引数を必要とします。「環境名」のところには、意味を与えたい環境名を書きます。そして、「開始文字列」と「終了文字列」のところには、環境名の意味となる文字列を書きます。

コマンドを定義するというのが文字列に名前を与えることだったのに対して、環境を定義するというのは、開始文字列と終了文字列のペアに名前を与えるということです。LaTeX は、\begin{環境名}というコマンドを、その環境名が与えられたペアのうちの開始文字列に置き換えます。そして同じように、\end{環境名}というコマンドを、その環境名が与えられたペアのうちの終了文字列に置き換えます。たとえば、

\newenvironment{booktitle}{『}{』}

というコマンドは、「『」と「』」という文字列のペアに対して booktitle という環境名を与えます。ですから、このコマンドよりも後ろに、

私の愛読書は、\begin{booktitle}意味を求めて\end{booktitle}という本です。

と書いたとすると、この文は LaTeX によって、

私の愛読書は、『意味を求めて』という本です。

に変換されます。

すでに定義されている環境を利用して新しい環境を作ることも可能です。それをしたいときは、利用したい環境の\begin コマンドを開始文字列の中に書いて、利用したい環境の\end コマンドを終了文字列の中に書きます。たとえば、

\newenvironment{quotelarge}{\begin{quote}\large}{\end{quote}}

というコマンドを書くことによって、quote 環境を利用して、引用文を大きな文字で出力する quotelarge という環境を定義することができます。

LaTeX ソースの例　newenvi.tex

```
\documentclass{jarticle}
\newenvironment{quotelarge}{\begin{quote}\large}{\end{quote}}
\begin{document}
彼は、『意味を求めて』の中で、
\begin{quotelarge}
この文は無意味である。
\end{quotelarge}
と述べています。
\end{document}
```

5.2.3 環境を再定義するコマンド

コマンドを定義するコマンドとコマンドを再定義するコマンドとが違っていたように、環境も、定義する場合と再定義する場合とでは、異なるコマンドを使う必要があります。

環境を再定義したいときは、\renewenvironment というコマンドを使います。

\renewenvironment コマンドの使い方は、\newenvironment コマンドとまったく同じです。1 個目の引数は、意味を変更したい環境名で、2 個目と 3 個目の引数は、新しい意味となる文字列です。

LaTeX ソースの例　renewen.tex

```
\documentclass{jarticle}
\renewenvironment{quote}
{\begin{tabular}{|c|}\hline}{\\\hline\end{tabular}}
\begin{document}
私は、\begin{quote}私は囲まれている\end{quote}と思いました。
\end{document}
```

5.3　引数

5.3.1　引数を持つコマンドの定義

この節では、引数を持つコマンドや環境を定義する方法について説明したいと思います。

引数を持つコマンドを定義したいときは、

```
\newcommand{コマンド名}[引数の個数]{文字列}
```

というように、\newcommand コマンドの 1 個目の引数と 2 個目の引数とのあいだにオプション引数を書きます。オプション引数の中には、定義されるコマンドが持つ引数の個数を書きます（引数は最大 9 個までです）。たとえば、

```
\newcommand{コマンド名}[4]{文字列}
```

というようなコマンドを書くことによって、4 個の引数を持つコマンドを定義することができます。

\newcommand コマンドの 2 個目の引数の中に、「仮引数」(parameter) と呼ばれる、#1、#2、･･･、#9 という文字列を書くと、それらの文字列は、1 個目の引数、2 個目の引数、･･･、9 個目の引数にそれぞれ置き換わります。たとえば、

```
\newcommand{\notbut}[3]{#1 は#2 ではなくて#3 です。}
```

というコマンドで\notbut というコマンドを定義したとしましょう。そののち、

```
\notbut{イモリ}{爬虫類}{両生類}
```

というコマンドを書いたとすると、このコマンドは、

```
イモリは爬虫類ではなくて両生類です。
```

という文を出力します。

\renewcommand コマンドを使ってコマンドを再定義する場合も、同じ方法で、引数を持つコマンドを作ることができます。

LaTeX ソースの例　argcomm.tex

```
\documentclass{jarticle}
\newcommand{\gtlarge}[1]{\textgt{\large #1}}
\begin{document}
文字列を\gtlarge{大きなゴシック体}で出力するコマンドを
定義してみました。
\end{document}
```

5.3.2　引数を持つ環境の定義

引数を持つ環境を定義したいときは、

```
\newenvironment{環境名}[引数の個数]{開始文字列}{終了文字列}
```

というように、\newenvironment コマンドの 1 個目の引数と 2 個目の引数とのあいだに、引数の個数（最大 9 個まで）をオプション引数として書きます。

コマンドの場合と同じように、#1 や #2 などの仮引数を開始文字列の中に書くと、それらの仮引数は引数に置き換わります。ただし、仮引数を終了文字列の中に書くことはできません。

\renewenvironment コマンドを使って環境を再定義する場合も、同じ方法で、引数を持つ環境を作ることができます。

LaTeX ソースの例　argenvi.tex

```
\documentclass{jarticle}
\newenvironment{titledquote}[1]
{\begin{quote}\textgt{#1}\\}{\end{quote}}
\begin{document}
その本を開くと、そこには、
\begin{titledquote}{桃太郎}
むかしむかしあるところに、おじいさんとおばあさんが（以下略）
\end{titledquote}
と書かれていました。
\end{document}
```

5.4 カウンター

5.4.1 カウンターとは何か

LaTeX は、「カウンター」(counter) と呼ばれるものを扱うことができるという機能を持っています。

カウンターは、1 個の整数を保持することのできる容器のようなものです。カウンターが保持している整数は、そのカウンターの「値」(value) と呼ばれます。カウンターの値は、変更することが可能です。

カウンターを使うことによって、何かの数を数えるということができます。LaTeX は、それぞれのページにページ数 (pagination) を出力したり、それぞれの節や項の見出しに番号を出力したり、 enumerate 環境の項目に番号を出力したりするために、カウンターを利用しています。

カウンターを利用することができるのは LaTeX だけではありません。LaTeX ソースを書く人も、新しいカウンターを作って、それを利用することができます。新しいカウンターを作ることを、カウンターの「定義」(definition) と呼びます。

5.4.2 カウンター名の作り方

カウンターは、「カウンター名」(counter name) と呼ばれる名前によって識別されます。カウンターを定義するためには、新しいカウンター名を作る必要があります。

カウンター名は、1 個以上の英字または日本語の文字を並べることによって作ります。たとえば、

raccoon　SVG　EtInArcadiaEgo　赤道儀

などは、正しいカウンター名です。

英字の大文字と小文字は区別されますので、 MUD と mud とは異なるカウンター名とみなされます。

5.4.3 カウンターを操作するコマンド

カウンターを定義したり、その値を変更したり、その値を出力したりしたいときは、次のコマンドを使います。

\newcounter　引数をカウンター名とするカウンターを定義して、それに対して 0 を値として設定する。たとえば、

```
\newcounter{namako}
```

というコマンドは、 namako というカウンター名によって識別されるカウンターを定義して、それに対して 0 を値として設定する。

\setcounter　1 個目の引数によって識別されるカウンターに対して 2 個目の引数を値として設定する。たとえば、

```
\setcounter{namako}{24}
```

というコマンドは、namako というカウンターに対して 24 を値として設定する。

\addtocounter　1 個目の引数によって識別されるカウンターの値を 2 個目の引数だけ増加させる。たとえば、 namako というカウンターの値が 24 だとするとき、

```
\addtocounter{namako}{3}
```

というコマンドは、 namako の値を 27 に増加させる。

\stepcounter　引数によって識別されるカウンターの値を 1 だけ増加させる。

\arabic　引数によって識別されるカウンターの値をアラビア数字 (Arabic numeral) で出力する。

\roman　引数によって識別されるカウンターの値をローマ数字 (Roman numeral) の小文字で出力する。

\Roman　引数によって識別されるカウンターの値をローマ数字の大文字で出力する。

\alph　引数によって識別されるカウンターの値を英字の小文字で出力する（カウンターの値は 26 以下でないといけない)。

\Alph　引数によって識別されるカウンターの値を英字の大文字で出力する（カウン

ターの値は 26 以下でないといけない)。

LaTeX ソースの例　counter.tex

```
\documentclass{jarticle}
\newcounter{blanknumber}
\newcommand{\blank}
{\stepcounter{blanknumber}(~\arabic{blanknumber}~)}
\begin{document}
太陽系の惑星の中で太陽にもっとも近いのは\blank で、もっとも
遠いのは\blank です。質量がもっとも大きいのは\blank で、
もっとも小さいのは\blank です。
\end{document}
```

参考文献

[Lamport,1994] Leslie Lamport, *LaTeX: A Document Preparation System, Second Edition*, Addison-Wesley, 1994, ISBN 978-0-201-52983-8. 邦訳（阿瀬はる美）、『文書処理システム LaTeX 2ε』、ピアソン・エデュケーション、1999、ISBN 978-4-89471-139-6。

[阿瀬,1994] 阿瀬はる美、『てくてく TeX』、アスキー出版局、1994、ISBN 978-4-7561-0222-5（上巻)、ISBN 978-4-7561-0223-2（下巻)。

[奥村,2007] 奥村晴彦、『LaTeX 2ε美文書作成入門・改訂第四版』、技術評論社、2007、ISBN 978-4-7741-2984-6。

[中野,1996] 中野賢、『日本語 LaTeX 2ε ブック』、アスキー出版局、1996、ISBN 978-4-7561-1667-3。

[本田,2005] 本田知亮、『LaTeX 2ε標準コマンドポケットリファレンス』、技術評論社、2005、ISBN 978-4-7741-2423-0。

索引