論文・レポート執筆のための

LaTeX 入門

本テキストの表記方法

コンソールアプリケーション上の操作

コンソールアプリケーション上のコマンド操作については，以下のような黒枠を用いる：

```
latex sample.tex
```

LaTeX の記述例

LaTeX の記述例については以下のような灰色の枠を用いる：

```
\documentclass[uplatex]{jsarticle}
```

このような記述の場合は，拡張子が.tex のテキストファイルとして VSCode やメモ帳などのテキストエディタ上に記述するものと考えてほしい．具体的にコマンドの入力例と出力例の両方を示す場合は，以下のように上部に入力例，下部に出力例を記述する：

```
$ax^2 + bx + c = 0$
```

$$ax^2 + bx + c = 0$$

多くの場合はこのような形式でコマンドの記述方法を紹介しているため，入力例と出力例を見比べることで記述方法を理解することができるだろう．

目次

第 1 章

LaTeX チュートリアル

まずは手始めに，手を動かして LaTeX を実行することで，LaTeX で文書を生成する流れを理解しよう．ここでは，最も単純な実行方法としてコマンドラインで操作する方法について説明する．

1.1 TeX ファイル

LaTeX で文書を作成する際には TeX 形式のファイルを用いる．拡張子は`.tex` である．TeX ファイルはテキストデータであるため編集にはテキストエディタを用いる．テキストエディタによっては，TeX をより便利に効率良く書けるような拡張機能が用意されているため，利用を検討してみると良い．編集した TeX ファイルを文書として閲覧できる状態にするためには PDF ファイルへ変換する必要がある．

1.2 TeX ファイルから PDF を生成する流れ

厳密には，「TeX」や「LaTeX」と呼ばれているものは複数ある．日本人の多くが用いている日本語対応した LaTeX を **pLaTeX** と呼び，それが UTF–8 に対応したものを **upLaTeX** と呼ぶのが正しい[*1]．これらの TeX は，レガシーな TeX と呼ばれている．

レガシーな TeX を用いる場合，図 1.1 のように TeX ファイルから直接 PDF ファイルが生成されることはなく，DIV ファイル (`.dvi`) と呼ばれるバイナリファイルが生成される．生成された DVI ファイルを `dvipdfmx` と呼ばれるツールを用いることで，PDF ファイルに変換できる．

[*1] 現実には，TeX や LaTeX と省略して呼ぶことが多いし，それでも伝わる．このテキストでも，省略してそのように読んでいる．

一方で，LuaLaTeX・XeLaTeX・pdfLaTeX のような，TeX ファイルから直接 PDF ファイルを生成できるものも存在する．これらの TeX は，モダンな TeX と呼ばれている．このテキストではレガシーな TeX の利用法について述べるが，ある程度 TeX を使えるようになれば移行するのはそこまで困難でないだろう．

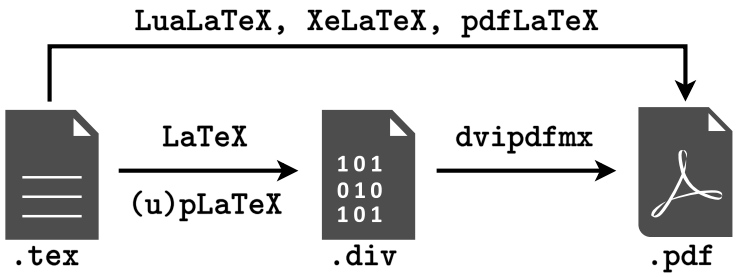


図 1.1　TeX ファイルから PDF ファイルを生成する流れ

なお，overleaf・CloudLaTeX や VSCode のようなエディタでレガシーな TeX 利用する場合，命令 1 つで PDF ファイルの生成を行えているように見えるが，内部ではこの手順で実行している．

1.3 コマンドラインを用いた PDF ファイル生成

この方法で実行することはあまり無いだろうが，コマンドラインを用いて TeX ファイルから PDF ファイルを生成する流れを具体的に紹介する．意味は理解できなくて良いので，サンプルとして以下の TeX ファイル (`sample.tex`) を作成してみよう：

```
\documentclass{jsarticle}
\begin{document}
  Hello, \TeX
\end{document}
```

まず，TeX ファイルから DVI ファイルを生成するために，以下のコマンドを実行する：

```
platex sample
```

これで，`sample.div`(とその他色々) が生成されただろう．生成されず，出力結果に「Error」のような文字列がある場合は記述が誤っている証拠であるので，修正する．正常に生成されていたら次は，DVI ファイルから PDF ファイルを生成するために，以下のコ

マンドを実行する：

```
dvipdfmx sample
```

これで，`sample.pdf` が生成されたハズである．

ここでは，最も単純な実行方法を紹介したが，実際には Overleaf・CloudLaTeX や VSCode を用いてボタン 1 つで実行することが殆どであるだろう．しかし，コマンド入力を用いる方法についても，緊急時のために知っておくと良い．

なお，ここでは 2 回のコマンドを実行して PDF ファイルを生成したが，TeX Live をインストールしている場合には `ptex2pdf` という先ほど処理を一括で行うコマンドが存在する[*2]：

```
ptex2pdf -l sample
```

[*2] upLaTeX を用いる場合にはオプション `-u` を追加する．

第 2 章

LaTeX の基礎用語

以下に示す用語は LaTeX の記法を説明したり，各々が Web や本で調べたりする際に頻出するものである．そのため，まずはこれらの用語について知っておいてほしい．

2.1 コマンド

LaTeX では，バックスラッシュ (\) から始まる文字列を**コマンド (命令)** と呼ぶ．コマンドには，コマンド名 (文字列) を指定して終わりのものもあれば，引数を指定するものもある．引数を指定する場合は，コマンド名以降に中括弧{}で囲んで指定する．例えば，架空のコマンド`\SampleCommand` に引数 `arg1` と `arg2` と `arg3` を指定する場合は，以下の通りに記述する：

```
\SampleCommand{arg1}{arg2}{arg3}
```

このように，LaTeX ではコマンドの引数を指定する際には**引数ごとに中括弧で囲む**ことに注意が必要である[*1]．また，コマンドには**オプション**と呼ばれる引数が存在することがある．オプションは，任意で指定する引数のことであり，必要に応じて追加で記述する．先程のコマンドにオプション `opt1` と `opt2` と `opt3` を指定する場合は，以下の通りに記述する：

```
\SampleCommand[opt1, opt2, opt3]{arg1}{arg2}{arg3}
```

オプションは，コマンド名と第 1 引数の中間に大括弧 `[]` で囲むことで指定し，区切りはカンマ (`,`) を用いる．通常の引数とは異なり，オプションは順不同である．

[*1] 引数に何も指定したくない場合は，空欄の中括弧を記述する．コマンドは，定義時に引数がいくつあるのかが決められているため，中括弧の数が合わないとエラーになる場合があることに注意．

2.2 環境

LaTeX では，以下のような記述が頻出する：

```
\begin{SampleEnv}
  何かしらの記述
\end{SampleEnv}
```

このような\begin で始まり\end で終わる箇所を，**環境**と呼ぶ．特に，上記のような場合は SampleEnv 環境と呼ぶ[*2]．

環境は，以下のように入れ子にすることも可能である：

```
\begin{SampleEnv1}
   \begin{SampleEnv2}
     何かしらの記述
   \end{SampleEnv2}
\end{SampleEnv1}
```

ただし，以下のように入れ子になっている\begin と\end の環境名が一致しない記述はエラーとなるので注意が必要である：

```
\begin{SampleEnv1}
   \begin{SampleEnv2}
     何かしらの記述
   \end{SampleEnv1}
\end{SampleEnv2}
```

2.3 コメント

マークアップ言語やプログラミング言語によってコメントの記号は異なるが，LaTeX ではパーセント (%) を用いる．改行するまで，% 以後の記述が全て実行に影響しなくなる：

[*2] 架空の名前であるため，実際にはこのような環境は存在しない．

```
表示される．% 表示されない
改行したので表示される．
```

表示される． 改行したので表示される．

なお，「%」 のように文章内にパーセントを記述したい場合は，\% と記述する．これ以外にも，LaTeX で特殊な意味を持つ記号については直接入力せずにコマンドで入力することがある (4.2 節を参照)．

2.4 ドキュメントクラス

LaTeX で文書を作成する際，1 行目に\documentclass コマンドを記述する．このコマンドは，**クラスファイル**と呼ばれる文書のレイアウトを予め定義してあるファイルを読み込むコマンドである．例えば，以下のように記述する：

```
\documentclass[a4paper, 11pt]{jsarticle}
```

ここで，第 1 引数として指定した jsarticle がクラスファイルである．その実態は，jsarticle.cls というテキストファイル[*3] であり，これを直接編集することは無い (勝手にいじると動作しなくなる可能性があるので注意)．現在，日本語でレガシーな LaTeX を用いる際に読み込むクラスファイルの多くは表 2.1 に示すものぐらいであるだろう．

[*3] https://github.com/texjporg/jsclasses/blob/master/jis/jsarticle.cls

表 2.1　よく用いるクラスファイル一覧

名前	用途	主な利用する場面
`jsarticle`	日本語論文用	通常のレポートや簡単な文書の規模であれば，これを用いる事が多い．基本的には，これに設定しておけば良いだろう．
`jsreport`	日本語報告書用	卒論や修論のような，章立てが必要な文書作成時に用いる．「第 1 章」のような書き方をしたい場合はこちらを用いる (4.4 節を参照)．
`jsbook`	日本語書籍用	書籍や製本を前提とした文書を作成するときに利用する．偶数・奇数ページで異なるページレイアウトを適用できる．
`beamer`	スライド作成用	LaTeX を用いてスライドを作成するときに利用する．数式を多用するスライドに有用である．

また，文書の見た目をオプション (**クラスオプション**) として指定することも可能である．

▶ 代表的なクラスオプション

- `a4paper`, `a5paper`, `b4paper`, `b5paper`

  用紙サイズを指定する．何も指定しないと `a4paper` になる (デフォルト)．指定できるサイズはこれ以外にも存在する．

- `9pt`，`10pt`，`11pt`，`12pt`，`14pt`，`17pt`，`12Q`，`14Q`，`10ptj`，`10.5ptj`，`11ptj`

  本文の文字サイズを指定する．[pt] 単位は欧文文字のもの，[ptj] 単位は和文文字のものである．[Q] 単位は級数と呼ばれる長さの単位であり，$1\,[\mathrm{Q}] = 0.25\,[\mathrm{mm}]$ である．これ以外にも指定できるサイズがあるが，全ての整数を指定できるとは限らない．

- `landscape`

  文書を横長にする．

- `twocolumn`

  文書を 2 段組にする．

- `titlepage`

  タイトルを独立したページにする．

- `fleqn`

数式を左寄せする (デフォルトでは中央寄せ)．数式の記述方法については第 8 章で述べる．

- `uplatex`

  upLaTeX を用いる場合には指定する．指定せずに upLaTeX を用いるとエラーになり[*4]，指定して pLaTeX を用いてもエラーとなる．なお，upLaTeX をコマンドラインで実行する際には以下のように入力する：

```
uplatex sample
```

  また，`ptex2pdf` を用いる場合はオプション`-u` を指定する：

```
ptex2pdf -u -l sample
```

2.5 document 環境とプリアンブル

LaTeX では，本文は全て **document 環境**の内部に記述する．一方で，ドキュメントクラスをはじめとする文書全般に関わる設定の記述や，外部パッケージ (第 3 章を参照) の読み込みについては `document` 環境の前に記述する．LaTeX では，`documentclass` コマンドから `document` 環境の前までの部分を**プリアンブル**と呼ぶ．つまり，LaTeX のソースコードの構造は簡略化すると以下のようになる：

```
\documentclass[uplatex,fleqn]{jsarticle}
% この部分がプリアンブル

\begin{document}
% ここに本文を記述する
\end{document}
```

今後の説明でも，プリアンブルに記述するコマンドについては「プリアンブルに以下を記述する」というような表現が多用されるため，その際に混乱しないようにしてほしい．

[*4] オプションに記載せずに，クラスファイルを `utarticle`, `utreport`, `utbook` にするという方法もある．

第 3 章

外部パッケージの利用

LaTeX は，世界中のいろいろな人が開発してくれている**パッケージ**と呼ばれる便利なマクロ集を利用することで真の力を発揮する．これにより，簡単なコマンドや環境で多種多様な表現を行えるようになる．知っておくと便利なパッケージに関しては，付録 A で紹介している．

3.1 CTAN

TeX に関する各種ファイルやソフトウェアを収集したオンラインリポジトリ (Web ページ) を **Comprehensive TeX Archive Network(CTAN)** [*1] と呼ぶ．Python における，PyPI に近いものであると考えると分かりやすいだろう．パッケージについても CTAN 上に保存されており，必要に応じてここからダウンロードすることが出来るが，その場合は後述する `tlmgr` を用いるのが便利である．

3.2 tlmgr

TeXLive Manager(`tlmgr`) は，TeX Live のパッケージ管理ツールである．Python における，pip に近いものであると考えると分かりやすいだろう．ローカル環境に TeXLive を用いて TeX をインストールしている場合は利用できる．`tlmgr` を用いれば，ローカル環境にイントールされていないパッケージをコマンドライン経由でインストールすることが可能である．ただし，我々が使うようなパッケージの多くは，予めインストールされている場合が殆どであるため，自前でインストールすることは稀であるだろう．

[*1] `https://www.ctan.org/`

▶ とりあえず，使えるか確認

利用している環境で tlmgr を使えるか確認してみよう．以下のコマンドを実行すれば，TeX Live のバージョンを確認できる (実行結果は筆者の環境のもの)：

```
tlmgr --version
> tlmgr revision 62273 (2022-02-28 09:52:17 +0100)
> tlmgr using installation: /usr/local/texlive/2022
> TeX Live (https://tug.org/texlive) version 2022
```

▶ パッケージインストール方法

例えば，パッケージ Ti*k*Z をイントールする場合は以下のコマンドを実行する[*2]：

```
tlmgr install tikz
```

▶ アップデート方法

インストール済みのパッケージ Ti*k*Z をアップデートする場合は，以下のコマンドを実行する：

```
tlmgr update tikz
```

全てのパッケージをアップデートしたい場合は，`tikz` の部分を`--all` にする：

```
tlmgr update --all
```

`tlmgr` そのものをアップデートする場合は，以下のコマンドを実行する：

```
tlmgr update --self
```

3.3 パッケージの導入

自分が編集中の TEX ファイル内にパッケージを導入するには，`\usepackage` コマンドを利用する．引数に使いたいパッケージ名を指定する．例えば，パッケージ Ti*k*Z を利用したい場合は，プリアンブルに以下のコマンドを記述する：

[*2] 実際には Ti*k*Z がインストールされていないことはまずないだろう．Ti*k*Z については付録 A で少しだけ解説する．

```
\usepackage{tikz}
```

なお，複数のパッケージで同一の名称のコマンドがある場合は，最後に導入したパッケージでのコマンドが優先される (上から下に読まれるので).

3.4 texdoc

通常，外部パッケージにはその仕様を詳細に記したドキュメンテーション (主に PDF) が用意されている．CTAN にアクセスすることでも閲覧は可能であるが，パッケージがローカル環境にインストールされている場合，ドキュメンテーションも一緒にインストールされているのでそちらを利用すると良い．`texdoc` コマンドを用いれば，ローカル環境で簡単にドキュメンテーションを開くことが出来る．例えば，パッケージ TikZ のドキュメンテーションを閲覧したい場合は，以下のコマンドを実行する：

```
texdoc tikz
```

実行すると，PDF ビューワが起動して，指定したパッケージのドキュメンテーションが開かれる．

第 4 章

基本的な文書記述法

ここから，本文の記述法について述べる．以後，特筆していない限りは document 環境内部に記述するものと考えてほしい．

4.1 タイトルの記述

文書の冒頭に表示されるタイトル・著者・日付の情報は，以下のようにプリアンブルに記述する：

```
\title{文書のタイトル}
\author{著者名}
\date{日付}
```

\date コマンドで今日の日付を指定したい場合は，\today コマンドを利用すると良い．そうすれば，「YYYY 年 MM 月 DD 日」のような表記で実行した時点での日付が表示される．

しかし，プリアンブルにこれを記述しただけでは本文中に反映されない．本文中に反映したい場合は，document 環境内の表示したい箇所に\maketitle コマンドを記述する．

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

4.2 本文の記述

日本語・アルファベット・数字に関してはそのまま入力すれば良い．キー入力可能である記号については，以下に示すもの以外はコマンド不要でそのまま入力できる：

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

これらの記号は，LaTeX で特殊な意味を持つ記号である．「<」は\textless，「>」は\textgreater，「\」は\textbackslash，「^」は\textasciicircum，「|」は\textbar，「~」は\texttildelow と入力する[*1]．それ以外の記号は\% のように記号前にバックスラッシュを付ければ良いだけである．

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

$ % & _ { }

段落については，1 行目に自動でインデントが付与される．段落を変えたい場合は，1 行空行を開ける．ただ改行しただけでは段落は変わらないので，エディタ内で自分が読みやすい位置で自由に改行することができる．

```
この部分は第１段落のままである．
空行で無い改行については同じ段落のままになるし，
勝手に改行されることはない．
自分が見やすい箇所で改行すると良い．

空行を開けたのでここから第２段落になる．
```

この場合，以下のような結果が得られる：

> この部分は第 1 段落のままである．空行で無い改行については同じ段落のままになるし，勝手に改行されることはない．自分が見やすい箇所で改行すると良い．
>
> 空行を開けたのでここから第 2 段落になる．

なお，強制的に改行を行うコマンドで「\\」というものがあるが，これを本文中の成形目的で乱用するのは推奨された書き方でないので注意してほしい．

[*1] 数式を記述する際にはコマンド不要である (第 8 章参照，バックスラッシュは例外)

4.3 文字の配置

特に指定しない場合，本文は左揃えとなる．これを中央揃えにしたい場合は `center` 環境，右揃えにしたい場合は `flushright` 環境を用いる．具体的には，以下のように記述する：

```
左揃え

\begin{center}
  中央揃え
\end{center}

\begin{flushright}
  右揃え
\end{flushright}
```

左揃え

中央揃え

右揃え

これらの環境は文字以外のもの (画像や表など) を中央揃え・右揃えにすることも可能である．

4.4 見出し

本テキストでも，上記のように見出しを設定している[*2]．LaTeX ではこのような見出しをコマンドで指定することで見出しの番号を自動で割り振ることができる．そのため，途中で見出しの番号が変わるような状況になったとしても手動でわざわざ番号を変え直す必要が無く，自動で変わってくれる．例えば，見出しは以下のように記述する：

```
\chapter{章見出しの名前を記述}
本文については,先述の通り書けば良い.

\section{節見出しの名前を記述}
章見出しのナンバリングが 1であれば,節見出しのナンバリングは 1.1になる.
```

[*2] ちなみに，本テキストの見出しの装飾は `titlesec` パッケージを利用してデフォルトのものから手を加えている．

見出しのコマンドは表 4.1 に示す通りである．特に，jsarcitle では\chapter コマンドが利用できないので注意が必要である．

表 4.1　見出しコマンド一覧

コマンド	役割	説明
\part	部見出し	「第 I 部」のように大きな区分を示す際に使用する．見出しのレベルとしては最上位であるが，使わなくても問題ない．
\chapter	章見出し	「第 1 章」のように章番号を付けて表示する．jsreport,jsbook で使用可能で，長めの文書ではこのコマンドから構成を始める．jsarticle では使用不可．
\section	節見出し	「1.1」のように章番号の下位に属する見出しを作成する．jsreport, jsbook では章の下位だが，jsarticle ではこのコマンドから構成を始める．
\subsection	小節見出し	「1.1.1」のように節の下位に属する見出しを作成する．
\subsubsection	小々節見出し	「1.1.1.1」のように小節のさらに下位の見出しを作成する．
\paragraph	段落見出し	番号は付かず，太字で表示される．短い補足説明や強調したいポイントを示すのに適している．
\subparagraph	小段落見出し	段落見出しよりさらに細かい単位で使われる．

4.5 箇条書き

4.5.1 通常の箇条書き

箇条書きは itemize 環境を用いる．この内部で各項目の先頭に\item コマンドを記述する：

```
\begin{itemize}
  \item C 言語
  \item Python
  \item Java
  \item Ruby
\end{itemize}
```

- C 言語
- Python
- Java
- Ruby

また，itemize 環境は 4 つまで入れ子にすることが可能である：

```
\begin{itemize}
  \item 第 1レベル
  \begin{itemize}
    \item 第 2レベル
    \begin{itemize}
      \item 第 3レベル
      \begin{itemize}
        \item 第 4レベル
      \end{itemize}
    \end{itemize}
  \end{itemize}
\end{itemize}
```

- 第 1 レベル
  - 第 2 レベル
    - 第 3 レベル
      - 第 4 レベル

4.5.2 番号付き箇条書き

番号付き箇条書きは enumerate 環境を用いる．項目の記述は itemzie 環境と同様である：

```
\begin{enumerate}
  \item 字句解析
  \item 構文解析
  \item 意味解析
  \item コード最適化
  \item コード生成
\end{enumerate}
```

1. 字句解析
2. 構文解析
3. 意味解析
4. コード最適化
5. コード生成

同様に，enumerate 環境も 4 つまで入れ子にすることが可能である：

```
\begin{enumerate}
  \item 第1レベル
  \begin{enumerate}
    \item 第2レベル
    \begin{enumerate}
      \item 第3レベル
      \begin{enumerate}
        \item 第4レベル
      \end{enumerate}
    \end{enumerate}
  \end{enumerate}
\end{enumerate}
```

1. 第 1 レベル
   (a) 第 2 レベル
      i. 第 3 レベル
         A. 第 4 レベル

4.6 注釈

これまでにも登場していた「*3」のような補足説明を記す注釈を記述するには`\footnote`コマンドを利用し，引数に補足説明を記述する：

```
AI\footnote{Artificial Intelligence}は,日本語では人工知能という.
```

AI[a]は，日本語では人工知能という．

[a] Artificial Intelligence

4.7 文字の変更

4.7.1 文字サイズ

文字サイズは，具体的な長さを数値で指定することも可能であるが，「本文サイズよりも大きくしたい」程度のことであれば，以下のようなコマンドで相対的に変更可能である：

```
普通のサイズ，{\Large 大きめのサイズ}
```

普通のサイズ，大きめのサイズ

文字サイズを変更したい範囲を中括弧で囲んだ上で，その括弧内部の左側にサイズ変更のコマンドを記述する．囲まないと，以下のように変更したい範囲外まで文字サイズが変わる恐れがあるので注意が必要である：

```
この部分は変えたくない.
\large この部分は大きくしたい.
この部分は変えたくない.
```

この部分は変えたくない．この部分は大きくしたい．この部分は変えたくない．

*3 これが注釈である．

```
この部分は変えたくない．
{\large この部分は大きくしたい．}
この部分は変えたくない．
```

この部分は変えたくない．{\large この部分は大きくしたい．}この部分は変えたくない．

文字サイズ変更のコマンドは表 4.2 に示す通りである．これらのサイズは，相対的なものであるので本文サイズ (\normalsize) に応じて異なる：

表 4.2　文字サイズ変更のコマンド

コマンド	表示例	コマンド	表示例
\tiny	サンプル	\scriptsize	サンプル
\footnotesize	サンプル	\small	サンプル
\normalsize	サンプル	\large	サンプル
\Large	サンプル	\LARGE	サンプル
\huge	サンプル	\Huge	サンプル

4.7.2 フォント変更 (簡易版)

簡単な書体変更方法

LaTeX で自分の好きなフォントをファイル単位で指定するのは，かなり複雑である[*4]．LuaLaTeX を用いれば簡単にフォントを設定できるので，「どうしてもこのフォントじゃないとダメ」というこだわりがある場合はそちらの利用を検討すると良い．

一方で，「明朝体をゴシック体に変換したい」というように書体の種類を変えることであれば簡単に行える．デフォルトの本文では欧文書体は Serif 体，和文書体は明朝体になっている．例えば，明朝体をゴシック体に変える場合は\textgt コマンドを用いる：

[*4] 欧文フォントであれば予め多数用意されているのだが，和文フォントはそうでない．

```
明朝体，\textgt{ゴシック体}
```

明朝体，ゴシック体

欧文フォントの種類

欧文フォントの種類 (書体) としては，Serif 体・Sans-Serif 体・タイプライタ体[*5]が利用可能である (表 4.3)．ウェイトについては，標準 (Medium) と太字 (Bold) が利用可能である (表 4.4)．シェープについては立体 (変形なし)・Italic・Oblique[*6]・Small Caps[*7]が利用可能である (表 4.5).

表 4.3　欧文フォントの種類

コマンド	表示例	説明
`\textrm{The quick brown fox}`	The quick brown fox	Serif 体 (デフォルト)
`\textsf{The quick brown fox}`	The quick brown fox	Sans-Serif 体
`\texttt{The quick brown fox}`	The quick brown fox	タイプライタ体

表 4.4　フォントのウェイト

コマンド	表示例	説明
`\textmd{The quick brown fox}`	The quick brown fox	標準 (デフォルト)
`\textbf{The quick brown fox}`	**The quick brown fox**	太字

[*5] ここで設定されるフォントは，幅が同じなので等幅 (モノスペース) フォントと呼ばれている．

[*6] Italic と Oblique のいずれも文字を斜めにした形状である．Italic は文字の形状が筆記体に近くなっており，Oblique は立体を斜めに変形した形状という違いがある．

[*7] 「Hello, World」のように，小文字の部分が，大きさが小さいまま形状だけ大文字になる．

表 4.5　フォントのシェープ

コマンド	表示例	説明
`\textup{The quick brown fox}`	The quick brown fox	立体 (デフォルト)
`\textit{The quick brown fox}`	*The quick brown fox*	Italic
`\textsl{The quick brown fox}`	*The quick brown fox*	Oblique
`\textsc{The quick brown fox}`	THE QUICK BROWN FOX	Small Caps

4.8 より細かいテクニック

4.8.1 引用符

本文中で引用符を用いる場合，ダブルクォーテーション (") で囲むと不格好になるので左側はアクサングラーブ[*8]2 つ (``)，右側はシングルクォーテーション 2 つ ('') で囲む必要があることに注意が必要である (以下の例だと文字の違いが分かりづらいかもしれない)：

```
"これはダメ"，``こっちにする''
```

"これはダメ"，“こっちにする”

適切に記述した場合，左側と右側で記号が異なっていることがわかるだろう．

4.8.2 空白

意図的に空白をあけたいとき，LaTeX では強制改行の記号や全角スペースを用いて力技であけるのは推奨されていない．もし空白を開けたい事情があるならば，`\hspace` や`\vspace` のようなコマンドを用いるべきである．`\hspace` は横方向に任意の長さの空白を開けるコマンドである．第 1 引数に単位付きで長さを指定する：

[*8] キーボードにもよるが，JIS 配列ではアットマーク (@) が印字されているキーを Shift キーと同時押しすれば入力できる．

```
横方向の\hspace{5cm}空白
```

横方向の　　空白

\vspace は縦方向に任意の長さの空白を開けるコマンドである[*9]：

```
縦方向の
\vspace{1em}
空白
```

縦方向の

空白

また，横方向に無制限の空白 (ページの右端まで) を開けるコマンドとして\hfill コマンドが存在する．これを活用すれば，1 行だけなら flushright 環境を用いなくても右寄せができる：

```
横方向の \hfill 空白
```

横方向の　　空白

同じ行に\hfill を複数回記述すると，空白が均等になる：

```
均等に \hfill 空白が \hfill 表示される
```

均等に　　空白が　　表示される

同様に，縦方向に無制限の空白 (ページの下側まで) を開けるコマンドとして\vfill コマンドも存在する．

[*9] 1 [em] は，\normalsize の全角文字 1 文字分の長さの相対値であると考えて構わない．

4.8.3 改ページ

強制的に次のページに切り替えるには，該当箇所で\newpage コマンドを用いる[*10]：

```
これは現在のページに表示される
\newpage
これは次のページに表示される
```

これにより，現在のページに文字や図などを配置できる余白があったとしても，このコマンド以下の内容は次のページに切り替わって表示される．

4.8.4 文字の装飾

ルビ

「輻輳(ふくそう)」のように漢字に対して読み方を明示したふりがなをルビと呼ぶ．LaTeX でルビを振る場合，pxrubrica パッケージを利用する：

```
\usepackage{pxrubrica}
```

ルビを振るには\ruby コマンドを利用する．第 1 引数に親文字 (漢字など) を指定し，第 2 引数にルビ文字を指定する．親文字が 2 文字以上である場合は，各文字の境界部にバーティカルバー (|) を入力する必要がある：

```
お\ruby{呪}{まじな}い，\ruby{魑魅魍魎}{ち|み|もう|りょう}
```

お呪(まじな)い，魑魅魍魎(ちみもうりょう)

「時雨(しぐれ)」のように，漢字によってはふりがなを文字ごとに分割せずに熟語自体に読みが与えられるものも存在する[*11]．そのような場合は，オプションに g を指定する：

```
\ruby[g]{五月雨}{さみだれ}
```

五月雨(さみだれ)

[*10] 同じ効果をもたらすコマンドは他にも存在する．
[*11] 熟字訓と呼ばれる．

圏点

圏点 (傍点) は日本語で文字を強調するために用いられる，横書きでは文字の上部に，縦書きでは文字の右側に付加される点 (・) のことである．先ほど紹介した pxrubrica パッケージには圏点を付けるコマンドが用意されている：

```
\kenten{圏点}
```

圏点

線

簡単に下線を引きたい場合は\underline コマンドを用いる：

```
\underline{下線の例}
```

下線の例

それ以外のバリエーションの線を引きたい場合は ulem パッケージを利用する：

```
\usepackage{ulem}
```

```
\uline{普通の下線}, \uuline{二重下線}, \sout{打ち消し線}, \uwave{波線}
```

普通の下線, 二重下線, ~~打ち消し線~~, 波線

文字の囲み

簡単に文字の周辺を囲みたい場合は\fbox コマンドを用いる：

```
テキスト内の一部だけ\fbox{枠で囲む}ことができる．
```

テキスト内の一部だけ 枠で囲む ことができる．

第 5 章

相互参照

相互参照は，LaTeX において非常に重要な機能の一つである．これを身につけることができれば，かなり効率的に執筆活動を行うことができるようになるだろう．

5.1 相互参照の大切さ

このテキスト内でも，「図 1.1」・「表 2.1」・「1.2 節」のように，文章中に図・表・節の番号を記している．このような箇所を**相互参照**と呼ぶ．TeX では，ページ・数式・章・節・図・表など自動で番号が割り振られるものに対して相互参照を簡単に行うことが可能である．

TeX では，ソースコード内に直接「図 1」というように具体的な数値を書き込むことは絶対にしてはならない．この書き方をしてしまうと，何らかの理由により図のナンバリングが変わってしまった場合に手作業で番号を修正しなければならなくなる．数箇所程度であれば負担は少ないだろうが，ソースコードが長くなってしまったり相互参照が増えたりしている場合，修正ミスが頻発する原因になる．そのため，以下に記す方法で相互参照を行ってほしい．

5.2 相互参照の方法

相互参照をする際，図・表などに対してユニークなラベルを割り当てる．ラベルは，図・表などに対して名札を付けるイメージを持つと分かりやすい (図 5.1).

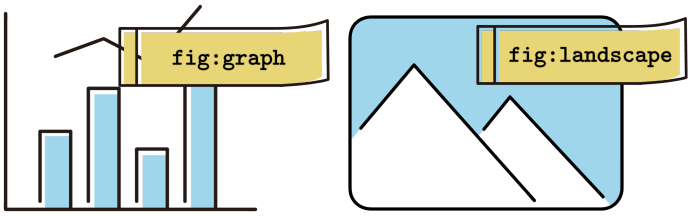


図 5.1　ラベルのイメージ

例えば，節見出しにラベルを付ける場合は以下のように記述する：

```
\section{サンプルテキスト} \label{sec:sample}
```

`\label` は番号を出力するコマンドの直後に記述する (改行は可)．なお，ラベル名は以下を守る必要がある：

- 「`\, {, }, %`」は利用できない
- 大文字と小文字は区別される
- 複数のラベルで同じ名前にすることは出来ない (ユニークである)

文章内で番号を呼び出す場合は，`\ref` コマンドを用いる：

```
詳細は\ref{sec:sample}節で述べる.
```

仮に，節番号が 1.1 であった場合には「詳細は 1.1 節で述べる.」と出力される．また，そのラベルに該当する部分のページ番号を参照したい場合は，`\pageref` コマンドを用いる：

```
詳細は\pageref{sec:sample}ページで述べる.
```

仮に，ページ番号が 18 であった場合には「詳細は 18 ページで述べる.」と出力される．

推奨されるラベル名のルール

基本的にラベル名については先述の条件を守れていれば問題ないのだが，管理のしやすさのために，表 5.1 に示すようにそのラベルが何を示すのかを明記することが推奨されている．例えば，図の場合は`\label{fig: 任意の名前}`のようにする．

表 5.1　推奨されているラベル名の命名規則

参照元	記法の例
図	`fig`
表	`tab, tbl`
数式	`eq, equ`
章	`chap`
節	`sec`
小節	`subsec`

5.3 実行時の注意点

相互参照を利用する際，一度 TEX を実行しただけでは想定した出力は得られない．例えば，「図 1 に示す」と表示されるところが，「図??に示す」というように番号部分が「??」となり参照がうまく反映されない．これは TEX の仕様上の問題[*1]であり，再び実行すると正しく反映される．

つまり，正しく相互参照を反映させるためには 2 回 TEX を実行する必要がある．利用しているエディタの設定次第では 1 回の命令だけで相互参照が適切に行えているように感じるものもあるが，内部では複数回 TEX を実行しているハズである．

[*1] 1 回目の実行で相互参照の管理を行うファイル (`.aux`) の内容が更新されるが，その時点では出力には反映されない．そのため，更新内容を出力に反映させるためにはもう一度実行が必要である．

第 6 章

画像の配置

レポート課題や学位論文では，画像を貼る際に図番号の参照やキャプションを正確に記述することが求められる．LaTeX を用いれば，それらを簡単に設定することが可能であり，後で変更することも容易に行える．

6.1 パッケージの導入

TeX で画像を貼る際，graphicx パッケージが必須となるためプリアンブルに以下のコマンドを記述する：

```
\usepackage[dvipdfmx]{graphicx}
```

また，必須ではないが float パッケージも読み込んでおくと良い：

```
\usepackage{float}
```

詳細は後述するが，このパッケージを読み込んでおくと画像を配置する位置として指定できる選択肢が増える．

6.2 画像の配置方法

6.2.1 1 枚の画像を配置

画像を配置するには figure 環境を用いる．この環境の内部に画像のパス・画像のタイトルのような画像に関する情報を以下にようにコマンドで記述する：

```
図\ref{fig:cherry}は筆者が 2021年に悠久山公園で撮影した桜の写真である：
\begin{figure}[H]
  \centering
  \includegraphics[width=0.7\linewidth]{spring.JPG}
  \caption{悠久山公園で撮影した桜}
  \label{fig:cherry}
\end{figure}
```

図 6.1 は筆者が 2021 年に悠久山公園で撮影した桜の写真である：

図 6.1　悠久山公園で撮影した桜

`\centering` コマンドを記述すると中央揃えになる[*1].

オプション

`figure` 環境のオプションには画像をどの位置に配置するのかを指定する．上記の例では `[H]` としているが，これは「必ずその位置に図を配置する」ことを表す[*2]．それ以外にも，ページの上部に配置するには `[t]`，ページの下部に配置するには `[b]`，できればその位置に配置するには `[h]`，独立したページに配置するには `[p]` と記述する．これらのオプションは `[htb]` のように配置されても良い条件を複数指定することも可能である[*3]．ただし，`[H]` は単独でしか指定できない．

[*1] `figure` 環境の内部で `center` 環境を入れ子にするのと同様の効果を得られる．
[*2] これを用いるのに `float` パッケージが必要である．
[*3] `h,t,b,p` の優先度で配置できる場所に配置するため順番は関係ない．

\includegraphics コマンド

画像ファイルの指定には\includegraphics コマンドを利用する．第 1 引数に画像ファイルの相対パス or 絶対パスを指定する．画像ファイルは JPG や PNG 形式以外にも，PDF 形式も指定できる．相対パスの場合，編集中の TEX ファイルからのパスである[*4]．

オプションには，画像の横幅・縦幅など，形状に関する情報を記載できる．例えば，[width=任意の長さ] と指定すれば，横幅を任意の長さに指定できる (縦横比は固定)．長さには [cm]・[in]・[pt] など，単位が必要である．[width=10cm] のように記述すれば良い．また，\linewidth と指定すると余白を除いた横幅の長さ[*5]になり，0.5\linewidth と指定すると\linewidth の 0.5 倍の長さになる．画像配置時にはこのコマンドを用いるのが便利である．

画像のタイトルと相互参照

「図 6.1　悠久山公園で撮影した桜」のような画像を説明するタイトル (キャプション) を記述するには\caption コマンドを利用する．このコマンドをを利用することで，図番号が自動で付与される．そのため，相互参照のためのラベル (\label) はこのコマンドの直後に記述する．

6.2.2 複数枚の画像を配置

複数枚の画像を横並びに配置するためには，minipage 環境を用いる．この環境は，ページを横方向に分割することが出来るものである．この環境を figure 環境内に複数記述して，それぞれの minipage 環境内で先述のような方法で画像を配置すれば良い．分割時，どれくらいの横幅にするのかを\begin{minipage}{横幅}のように第 2 引数に指定する．このとき，分割した minipage の合計が元々のページの横幅を超えてしまうと，はみ出してしまい警告が出るため注意が必要である．また，この環境内での\linewidth は minipage の幅になる．minipage 環境を用いて 2 枚の画像を横並びに配置した例を次ページに示す：

[*4] つまり，sample.tex と同じディレクトリに figures というディレクトリがあり，その中に graph.png という画像ファイルがあった場合，相対パスは figures/graph.png となる．また，\graphicspath コマンドを用いることで，画像ファイルの保存されているディレクトリを指定することも可能である．

[*5] 利用している箇所によって長さが異なる．例えば，例示に利用している枠内と本文の箇所では長さが異なっている．

```
\begin{figure}[H]
  \centering
  \begin{minipage}{0.4\linewidth}
    \centering
    \includegraphics[width=\linewidth]{summer.JPG}
    \caption{長岡まつり大花火大会}
    \label{fig:fireworks}
  \end{minipage}
  \hspace{10pt}
  \begin{minipage}{0.4\linewidth}
    \centering
    \includegraphics[width=\linewidth]{autumn.JPG}
    \caption{悠久山公園で撮影した紅葉}
    \label{fig:leaf}
  \end{minipage}
\end{figure}
```

図 6.2　長岡まつり大花火大会

図 6.3　悠久山公園で撮影した紅葉

横並びにした際に画像間の間隔が窮屈に感じる場合は，`\hspace` コマンドを用いて調整すると良い．

第 7 章

表の作成

LaTeX で表を作成するのは慣れるまではやや大変であるが，今後の執筆活動で避けては通れないハズなので頑張って身につけてほしい．

7.1 表の配置方法

表を配置するには，table 環境を用いる．書き方のルールは figure 環境と同様であるが，表の場合はキャプションを上に記述する．

```
表の表示例を表\ref{tab:table_sample}に示す：
\begin{table}[H]
  \centering
  \caption{表のキャプション}
  \label{tab:table_sample}
  ここに表を記述する (やり方は後述)
\end{table}
```

表の表示例を表 7.1 に示す：

表 7.1　表のキャプション

ここに表を記述する (やり方は後述)

記述の方法は画像の配置と同様の考えである．横並びについても minipage 環境を用いれば表現できる．

7.2 表組み

ここから，表の作成方法を述べる．表の作成には tabular 環境を用いる．表の作成例を以下に示す：

```
\begin{table}[H]
  \centering
  \caption{製品情報}
  \label{tab:seihin}
  \begin{tabular}{c||r|r}
    \hline
    製品名 & 価格 [円] & 在庫数 \\
    \hline \hline
    ノートパソコン & 120,000 & 25 \\
    スマートフォン & 85,000 & 50 \\
    ディスプレイ & 30,000 & 35 \\
    マウス & 10,000 & 120 \\
    キーボード & 5,000 & 60 \\
    \hline
  \end{tabular}
\end{table}
```

表 7.2　製品情報

製品名	価格 [円]	在庫数
ノートパソコン	120,000	25
スマートフォン	85,000	50
ディスプレイ	30,000	35
マウス	10,000	120
キーボード	5,000	60

tabular 環境の引数 (列指定)

\begin{tabular}{c||r|r}の{c||r|r}の部分で，列数・各列の揃え方・罫線の有無を指定する．これを列指定と呼ぶ．それぞれの列について，c は中央揃え，l は左揃え，r は右揃えを表す．これらは列の数だけ指定する．つまり，1 列目を左揃え，2 列目を中央揃え，3 列目を右揃えにしたい場合は{lcr}と記述すれば良い．

バーティカルバー (|) を記述すると，その列の間に罫線が表示される．つまり，{lcr}に対し 1 列目と 2 列目の間だけ罫線を指定したい場合は{l|cr} と記述すれば良い．||と

記述すれば罫線が 2 重になる．

セルの記述方法

横方向のセルの区切りとして，アンパサンド (&) を記述する．セルの結合をしない限り，1 行あたりのアンパサンドの数は「列数 − 1」になるハズであり，数が合わない場合はエラーとなる．

改行する場合は強制改行の記号「//」を用いる．その際，横方向に罫線を引きたい場合は\hline コマンドを記述する．このコマンドを連続で記述すれば二重になる．

セルの結合

横方向にセルを結合する場合，\multicolmun コマンドを用いる．該当箇所のセルで\multicolmun{結合するセル数}{列指定}{内容}のように記述する：

```
\begin{table}[H]
  \centering
  \caption{セルを結合した例}
  \label{tab:invoice}
  \begin{tabular}{|c|r|r|} \hline
    \multicolumn{3}{|c|}{請求書} \\ \hline
    品名 & 単価 [円] & 数量 \\ \hline
    バナナ & 50 & 3 \\
    トマト & 80 & 9 \\
    じゃがいも & 30 & 5 \\ \hline
    \multicolumn{2}{|r|}{合計} & 1020円 \\ \hline
  \end{tabular}
\end{table}
```

表 7.3　セルを結合した例

請求書		
品名	単価 [円]	数量
バナナ	50	3
トマト	80	9
じゃがいも	30	5
	合計	1020 円

第 8 章

数式

LaTeX を利用する大きなメリットの一つに多種多様な表現の数式を綺麗に描画できることがある．最初のうちは知らないコマンドが多く存在するため，入力に時間がかかってしまうかもしれないが，多用するコマンドについては自然と覚えるハズである．

8.1 数式表示の準備

8.1.1 インライン数式表示

例えば，「求めたい解は $x = 1$ である」のように本文中に数式を表示したい場合は，該当部分をドルマーク (`$`) で囲む：

```
求めたい解は$x = 1$である．
```

求めたい解は $x = 1$ である．

このような数式表示方法を**インライン数式 (textstyle)** と呼ぶ．

8.1.2 別行立て数式表示

例えば，

$$\frac{dx}{dt} = x \tag{8.1}$$

のように，行を変えて数式を表示したい場合は，`equation` 環境を用いる：

```
式 (\ref{equ:diff})は,変数分離形の微分方程式である:
\begin{equation}
  \frac{dx}{dt} = x
  \label{equ:diff}
\end{equation}
```

式 (8.2) は，変数分離形の微分方程式である：

$$\frac{dx}{dt} = x \tag{8.2}$$

equation 環境を用いると，自動で数式番号が付与される．「式 (8.2)」のように相互参照のためにラベルを付与する場合は，環境内に\label コマンドを記述する．数式番号が不要な場合は，以下のいずれかの方法を用いれば良い (これ以外にもある)：

```
\begin{equation*}
  \frac{dx}{dt} = x
\end{equation*}

\[
  \frac{d^2 x}{dt^2} = -x^2
\]
```

$$\frac{dx}{dt} = x$$

$$\frac{d^2x}{dt^2} = -x^2$$

このような数式表示方法を**別行立て数式 (displaystyle)** と呼ぶ．

8.1.3 数式表示に (ほぼ) 必須となるパッケージ

多用しやすい数学記号の中には，以下のパッケージを導入しなければ利用できないものも多いため，予めプリアンブルに以下のコマンドを記述しておくと良い：

```
\usepackage{amsmath, amssymb}
```

以後の記述でも，これらのパッケージは導入されているものとみなして進める．

8.2 代表的な数式の記述法

数式の記述法を具体例とともに説明する．具体例では本文中では説明していないが，知っておくと便利な記号のコマンドも利用している．その部分に関してはソースコードと対応させながら理解してほしい．

8.2.1 数字・アルファベット・ギリシャ文字

▶ 数字・アルファベット

数字・アルファベットについては通常通りに入力すれば良い．数式中では，アルファベットはイタリック体になる：

```
$abc123$
```

$abc123$

なお，意図的に数式内に立体のアルファベットを表示したい場合は，Serif 体の場合は`\mathrm` コマンド，Sans-Serif 体の場合は`\mathsf` コマンドを利用する：

```
$\mathrm{hoge}, \mathsf{fuga}$
```

$\mathrm{hoge}, \mathsf{fuga}$

▶ ギリシャ文字

ギリシャ文字については，表 8.1 に示すようなコマンドを用いて入力する．なお，いずれもインライン数式 or 別行立て数式として利用しなければならない．そのため，本文中に使う場合はドルマークで囲む．

表 8.1　ギリシャ文字のコマンド

α	\alpha	β	\beta	γ	\gamma	δ	\delta
ϵ	\epsilon	ε	\varepsilon	ζ	\zeta	η	\eta
θ	\theta	ϑ	\vartheta	ι	\iota	κ	\kappa
λ	\lambda	μ	\mu	ν	\nu	ξ	\xi
o	o	π	\pi	ϖ	\varpi	ρ	\rho
ϱ	\varrho	σ	\sigma	ς	\varsigma	τ	\tau
υ	\upsilon	ϕ	\phi	φ	\varphi	χ	\chi
ψ	\psi	ω	\omega	Γ	\Gamma	Δ	\Delta
Θ	\Theta	Λ	\Lambda	Ξ	\Xi	Π	\Pi
Σ	\Sigma	Υ	\Upsilon	Φ	\Phi	Ψ	\Psi
Ω	\Omega						

8.2.2 四則演算・指数・添字

四則演算

加法・減法については，通常のプログラミング時と同じような方法で入力すれば良い：

```
$1 + 1 = 2, 5 - 2 = 3$
```

$1 + 1 = 2, 5 - 2 = 3$

一方で，乗算・除算については，記号をコマンドで入力する．キー入力方法が存在しないような記号はコマンドを用いることが多い：

```
$2 \times 3, 2 \cdot 3, 10 \div 5$
```

$2 \times 3, 2 \cdot 3, 10 \div 5$

指数

指数については，キャレット (^) で表す．この記号を入力することで，以降の 1 文字を上付き文字とすることが可能である：

```
$2^6 = 64, 2^10$
```

$2^6 = 64, 2^10$

2 文字以上の上付き文字の場合，`$2^10$`と入力すると 2^10 と表示されるため，想定とは異なる結果となっている．2 文字以上を上付き文字にしたい場合は中括弧で囲む必要がある．TEX における，中括弧は「囲んだ部分を 1 つのカタマリとみなす」という意味になると覚えておいてほしい[*1](以後も多用される)：

```
$2^{10} = 1024$
```

$2^{10} = 1024$

添字

添字については，アンダーバー (_) で表す：

```
$a_1, a_2, \ldots, a_{n-1}, a_n$
```

$a_1, a_2, \ldots, a_{n-1}, a_n$

添字として，$x_{\max}$ のように，添字に数値や変数以外の文字列を指定することがよくある．このような場合，以下のような記述では「max」がイタリック体になり，不格好である (これは $m \times a \times x$ になる)：

```
$x_{max}$
```

x_{max}

そのため，数式内で立体の Serif 体を表示したい場合は先述した`\mathrm` コマンドを用いるべきである：

[*1] 引数を指定する時に中括弧で囲んだのも 2 文字以上の文字列を指定するためである．

```
$x_{\mathrm{max}}$
```

x_{max}

8.2.3 分数・平方根・数学関数

分数

分数は，`\frac` コマンドを用いる．第 1 引数に分子，第 2 引数に分母を指定する：

```
\[
 \frac{1}{2}, \frac{n - 1}{2}
\]
```

$$\frac{1}{2}, \frac{n - 1}{2}$$

上記では別行立て数式で示したが，同様のコマンドをインライン数式で入力すると，$\frac{1}{2}$ のように分数のサイズが文字サイズに合わせて小さくなる．このような場合に別行立て数式と同様のサイズにしたい場合は，`\displaystyle` コマンドを用いる：

```
コイントスをして表が出る確率は$\displaystyle\frac{1}{2}$である．
```

コイントスをして表が出る確率は $\displaystyle\frac{1}{2}$ である．

分数以外のもこのようなコマンドは存在するが，いずれの場合でもこの方法を用いれば良い．なお，分数については`\dfrac` コマンドが存在するので，そちらを利用するとより簡潔に記述できる：

```
$\dfrac{1}{2}$
```

$\dfrac{1}{2}$

平方根

平方根は`\sqrt`コマンドを利用する：

```
$\sqrt{2} \approx 1.414$
```

$\sqrt{2} \approx 1.414$

累乗根を表現する場合はオプションに任意の数値や文字を指定する：

```
$\sqrt[3]{8} = \sqrt[3]{2^3} = 2$
```

$\sqrt[3]{8} = \sqrt[3]{2^3} = 2$

数学関数

▶ 三角関数等

立体の Serif 体で関数名を記すものについては，予め専用のコマンドが用意されていることが殆どである．三角関数の場合も，以下のようにそのまま関数名をコマンドとして記述する：

```
$\sin\theta, \cos\theta, \tan\theta$
```

$\sin\theta, \cos\theta, \tan\theta$

添字のときと同様に，コマンドを利用せずにイタリック体で関数名を表示するのは不格好であるので注意が必要である[*2]：

```
$sin\theta$ % これはダメ
```

$sin\theta$

双曲線関数も同様である：

[*2] コマンドを用いると，立体になるだけでなくコマンドの直後に空白が入る．

```
$\sinh x, \cosh x, \tanh x$
```

$\sinh x, \cosh x, \tanh x$

逆三角関数も同様である：

```
$\arcsin x, \arccos x, \arctan x$
```

$\arcsin x, \arccos x, \arctan x$

▶ 指数関数

ネイピア数を用いる場合は，これまでの内容で表現できる：

```
$e^{i\pi} = -1$
```

$e^{i\pi} = -1$

exp を使う場合はそのまま関数名をコマンドとして記述する：

```
$\exp(i\theta) = \cos\theta + i\sin\theta$
```

$\exp(i\theta) = \cos\theta + i\sin\theta$

▶ 対数関数

log はこれまでと同様に関数名をコマンドとして記述し，底については下付き文字として記述する：

```
$\log_{a} b = c \Leftrightarrow a^c = b$
```

$\log_{a} b = c \Leftrightarrow a^c = b$

自然対数も同様である：

```
$\ln e = 1$
```

$\ln e = 1$

8.2.4 集合論

集合の記法

集合に用いる中括弧を表示するためには，\{や\}のようにコマンドとして記述する．外延的記法については，以下のように記述する：

```
$\{1, 2, 3, 4, 5\}$
```

$\{1, 2, 3, 4, 5\}$

内包的記法については，以下のように記述する：

```
$\{ x \mid x^2 - 5x + 6 = 0 \}$
```

$\{ x \mid x^2 - 5x + 6 = 0 \}$

縦線として\mid というコマンドを用いたが，このコマンドを用いると左右に空白が入るのでこちらを用いると見た目が良くなる．一方で，$|-2|$ や $|A|$ のように絶対値や行列式を表す場合は，バーティカルバー (|) をキー入力すると良い．

帰属・包含関係

帰属関係については，以下のように記述する：

```
$x \in \mathbb{R}$
```

$x \in \mathbb{R}$

\mathbb コマンドは，フォントを黒板太字 (Blackboard Bold) にするコマンドである．また，「属さない」ことを明記する記号 $\notin$ は\notin である．

包含関係については，以下のように記述する：

```
$A \subset B$
```

$A \subset B$

集合演算

和集合・積集合は以下のように記述する：

```
$A \cup B, A \cap B$
```

$A \cup B, A \cap B$

補集合は以下のように記述する：

```
$\overline{A \cap B} = \overline{A} \cup \overline{B}$
```

$\overline{A \cap B} = \overline{A} \cup \overline{B}$

8.2.5 解析学

極限

極限は以下のように記述する：

```
\[
  \lim_{x\to\infty} \frac{1}{x} = 0
\]
```

$$\lim_{x\to\infty} \frac{1}{x} = 0$$

このコマンドも，$\lim_{x\to\infty} \frac{1}{x}$ のようにインライン数式では表示が異なるので注意が必要である．

微分

▶ 常微分

常微分については，ここまでの内容で表現可能である：

```
\[
 \frac{df}{dx} = \lim_{h\to 0} \frac{f(x + h) - f(x)}{h}
\]
```

$$\frac{df}{dx} = \lim_{h\to 0} \frac{f(x + h) - f(x)}{h}$$

▶ 簡易的な表記

プライムを用いたり，ドットを用いたりして微分を表現する方法もある：

```
$f'(x) = 2x, a(t) = \dot{v}(t) = \ddot{x}(t)$
```

$f'(x) = 2x, a(t) = \dot{v}(t) = \ddot{x}(t)$

▶ 偏微分

偏微分特有の記号 ∂ は`\partial` と入力する：

```
\[
 \frac{\partial f}{\partial x} = 2xy
\]
```

$$\frac{\partial f}{\partial x} = 2xy$$

大型演算子

数列の和を表す演算は，以下のように記述する：

```
\[
  \sum_{i = 1}^{n} i = \frac{n(n-1)}{2}
\]
```

$$\sum_{i=1}^{n} i = \frac{n(n-1)}{2}$$

同様に，数列の積を表す演算は，以下のように記述する：

```
\[
  \prod_{i = 1}^{n} i = n!
\]
```

$$\prod_{i=1}^{n} i = n!$$

積分

▶ 定積分

定積分は，以下のように記述する：

```
\[
  \int_{a}^{b} f(x)\, dx = F(b) - F(a)
\]
```

$$\int_{a}^{b} f(x)\, dx = F(b) - F(a)$$

コマンド「`\,`」は空白を開けるコマンドである．括弧と dx の間に僅かであるが空白が空いている方が見栄えが良くなる．なお，不定積分については下付き文字・上付き文字を記述しなければ良いだけである．

▶ 重積分

重積分は，以下のように記述する：

```
\[
  \iint_{S} f(x, y)\, dxdy
\]
```

$$\iint_S f(x,y)\,dxdy$$

積分区間が具体的に示されている重積分は，以下のように記述する：

```
\[
  \int_{0}^{1} \!\!\! \int_{0}^{y} x^2\, dx dy
\]
```

$$\int_0^1 \!\!\! \int_0^y x^2\,dxdy$$

コマンド「\!」は空白を狭めるコマンドである．これを書かないと，インテグラル同士の空白が空きすぎてしまい不格好になる．

8.2.6 線形代数

▶ 行列

行列の表現方法は複数あるが，pmatrix 環境を用いるのが一番簡潔であるだろう：

```
\[
  \begin{pmatrix}
    a & b & c \\
    d & e & f
  \end{pmatrix}
\]
```

$$\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}$$

成分は表と同じ感覚で記述する．また，bmatrix 環境を用いれば行列の括弧が大括弧になる：

```
\[
  \begin{bmatrix}
    a & b & c \\
    d & e & f
  \end{bmatrix}
\]
```

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

同様に，行列式は vmatirx 環境を用いる：

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

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

▶ ベクトル

矢印表記のベクトルで 1 文字の小文字であれば\vec コマンドを用い，1 文字以上の大文字であれば\overrightarrow コマンドを用いる：

```
$\vec{v}, \overrightarrow{A}, \overrightarrow{OA}$
```

$\vec{v}, \overrightarrow{A}, \overrightarrow{OA}$

太字表記のベクトルについては，bm パッケージを導入した上で\bm コマンドを利用すると良い．プリアンブルに以下のコマンドを記述し，利用する：

```
\usepackage{bm}
```

```
$\bm{F} = 2\bm{i} - 4\bm{j} + 3\bm{k}$
```

$$\boldsymbol{F} = 2\boldsymbol{i} - 4\boldsymbol{j} + 3\boldsymbol{k}$$

成分表示については，pmatrix 環境を用いれば良い：

```
\[
  \bm{a} =
  \begin{pmatrix}
    1 \\ 2
  \end{pmatrix},
  \bm{b} =
  \begin{pmatrix}
    3 & 4
  \end{pmatrix}
\]
```

$$\boldsymbol{a} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 3 & 4 \end{pmatrix}$$

8.3 より綺麗に数式を表示するテクニック

8.3.1 複数行の式を揃える

方程式の解を導出する過程や式変形など，複数行に一連の数式を書きたい場合，イコールで揃えたほうが見た目が良くなる．そのような場合には，align 環境を用いる：

```
\begin{align}
  x - 4 &= 3x + 6 \\
  -2x &= 10 \nonumber \\
  x &= -5
\end{align}
```

$$x - 4 = 3x + 6 \tag{8.3}$$
$$-2x = 10$$
$$x = -5 \tag{8.4}$$

align 環境内では，揃えたい記号の前にアンパサンド (&) を記述し，改行したい位置で\\を記述する．特に設定しない限りすべての行で数式番号が付与されるため，一部の行で不要な場合は\nonumber コマンドを使い，すべての行で不要な場合は環境名をalign*にすれば良い．

8.3.2 括弧の大きさを調整する

例えば，以下のような記述では括弧の大きさが小さすぎるため不格好になってしまう：

```
\[
  (1 + \frac{1}{x})^n
\]
```

$$(1 + \frac{1}{x})^n$$

このような場合に括弧の大きさを自動で調整するコマンドとして，\left と right がある．これらのコマンドを括弧の手前に記述する：

```
\[
  \left ( 1 + \frac{1}{x} \right )^n
\]
```

$$\left (1 + \frac{1}{x} \right)^n$$

括弧類であれば，中括弧や大括弧などでも同様に利用可能である．ただし，これらのコマンドは**必ず 2 つとも記述しなければならない**ことに注意が必要である．片方のみだとエラーとなってしまうため，以下のように左側だけ括弧を使いたい場合は，`\right .` のように記述する：

```
\[
  \left \{
    \begin{aligned}
      &x + y = 2 \\
      &2x - 3y = 1
    \end{aligned}
  \right .
\]
```

$$\left \{ \begin{aligned} &x + y = 2 \\ &2x - 3y = 1 \end{aligned} \right .$$

`aligned` 環境は，`equation` 環境などの数式環境内で `align` 環境の記法を使いたい時に用いる．

8.3.3 場合分け

ReLU 関数や符号関数のように，値によって場合分けして表記したい場合は，`cases` 環境を用いれば良い：

```
\[
  f(x) =
  \begin{cases}
    0 & (x < 0) \\
    x & (x \geqq 0)
  \end{cases}
\]
```

$$f(x) = \begin{cases} 0 & (x < 0) \\ x & (x \geqq 0) \end{cases}$$

8.3.4 最適化問題の定式化

定式化の表記については，`alinged` 環境内で整列したい位置に `&` を記述することでキレイに揃えることができる．

```
\[
  \begin{aligned}
    & \mathrm{maximize} & f(x) \\
    & \mathrm{subject~to} & g(x) \\
    & & h(x)
  \end{aligned}
\]
```

$$\begin{aligned} & \mathrm{maximize} & f(x) \\ & \mathrm{subject\ to} & g(x) \\ & & h(x) \end{aligned}$$

第 9 章

参考文献の記載法

LaTeX を用いれば，参考文献についても効率的に記述することができるようになる．また，それを手助けする文献管理ツールである BibTeX と併用すると更に効率が上がる．

9.1 TeX ファイル内に直接記述する

TeX ファイル内に参考文献リストを直接書き込む場合，thebibliogtraphy 環境を用いる．引数として，文献番号が 1 桁の場合は{9}，2 桁の場合は{99}，3 桁の場合は{999}のように指定する必要がある．数が不明な場合は多めの桁数にしておけば良い．各種文献の情報は，この環境内で\bibitem コマンドを用いて箇条書きと同じような方法で記述する．このコマンドの引数にユニークな参照名を記述する (これが後の参照で必要になる)．

```
\begin{thebibliography}{9}
  \bibitem{tex} 奥村晴彦・黒木裕介，‘‘LaTeX2$\varepsilon$美文書作成入門’’,技
      術評論社，2017
\end{thebibliography}
```

本文中で文献を参照する場合は，\ref コマンドではなく\cite コマンドを利用する：

```
\cite{tex}
```

9.2 BibTeX を用いる

参考文献の量が多くなってくる場合は，先程のような方法で文献情報を手打ちして記述するのは非効率である．そのような場合は BibTeX と呼ばれる文献管理ツールを用いるのが便利である．

9.2.1 文献情報の記述法

文献情報は拡張子が.bib のファイルに記述する．実用上はこれから述べる文献情報の書き方のルールに従ってタイトルや著者名などを手打ちすることは少なく，Web 上で配布されている BibTEX 形式の文献情報をコピー・アンド・ペーストすることが殆どである．

例えば，文献情報 (ここでは書籍) は以下のように記述する：

```
@book{2017latex,
  title={LaTeX2$\varepsilon$美文書作成入門},
  isbn={9784774187051},
  url={https://books.google.co.jp/books?id=xIlPMQAACAAJ},
  year={2017},
  publisher={技術評論社}
}
```

@book の部分をエントリと呼び，記載する文献の種類に応じて異なる．エントリの一例は表 9.1 に示すとおりである．エントリに応じてフィールドと呼ばれるタイトルや著者名など，必須で記載しなければならない項目や任意で記載できる項目が異なる．フィールドはこのエントリ内に「フィールド名={内容}」のような形式でカンマ区切りで記述する．エントリ名の直後にある 2017latex の部分は後ほど文献を相互参照するためのラベルである．

表 9.1　エントリの一例

エントリ	用途	必須フィールド
@book	書籍	title, author or editor, publisher, year
@article	学術論文	author, title, journal, year, volume, pages
@inproceedings	学会論文	author, title, booktitle, year
@masterthesis	修士論文	author, title, school, year
@misc	その他（Web 記事など）	author or editor, title, howpublished, year

9.2.2 TEX ファイルの記述法

文献情報は.bib ファイルに記載するため，実際の文書で表示したいときはこのファイルを読み込む必要がある．例えば，reference.bib を読み込む際は表示したい位置に以下のように記述する：

```
\bibliographystyle{junsrt}
\bibliography{reference.bib}
```

\bibliographystyle コマンドは文献情報の表示方法を指定するコマンドであり，junsrt は引用順に文献を並べる方法である．

また，文献を相互参照する場合は先程と同様に\cite コマンドを利用する：

```
\cite{2017latex}
```

9.2.3 実行時の注意点

相互参照のときと同様に 1 回の実行だけでは文献情報は反映されず，BibTEX を用いる場合は合計 4 回のプログラムの実行が必要である．例えば，sample.tex の実行を upLATEX と upBibTEX[*1]を用いてコマンドラインで行う場合は，次の順番で 4 回コマンドを実行する (同じコマンドも複数回実行する) [*2]：

```
ptex2pdf -u -l sample
upbibtex sample
ptex2pdf -u -l sample
ptex2pdf -u -l sample
```

また，文献情報を記載して読み込んだとしても，**文書内で 1 箇所も文献の参照を行っていない場合はエラーとなる**ため，注意が必要である．

[*1] 日本語・Unicode 対応した BibTEX.
[*2] 実用上は VSCode の設定などで 1 回の命令で一連の処理を自動化することが多い．

第 10 章

自作コマンド作成法

実は，ここまで色々と利用してきたコマンドは自分独自に作成することも可能である．

10.1 自分オリジナルのコマンドを作成

同じ内容を何回も記述するなら，自作のコマンドを作成するのが便利である．例えば，図の相互参照を行うには「図\ref{fig:sample}」のように記述していたが，毎回「図」と記述してから相互参照を行うのはやや面倒である．そこで，相互参照に用いるラベルだけを指定することで「図」まで表示する自作コマンド\figref を以下のように作成した (プリアンブルに記述)：

```
\newcommand{\figref}[1]{\figurename\ref{#1}}
```

新規でコマンドを作成する際には\newcommand コマンドを用いる．具体的な記述法としては，\newcommand{新規コマンド名}[引数の数]{コマンドの中身} のように記述する．引数が不要な場合はオプションは不要である．コマンドの中身の部分で，第 1 引数は「#1」，第 2 引数は「#2」のように記述すれば良く，最大 9 個まで引数を指定できる．実際に自作したコマンドを用いると，実行時にそのコマンドの部分が第 2 引数のコマンドの中身に置き換わり，#1 のような部分も指定した引数に置き換わると考えれば良い．つまり，以下のようなイメージである：

$$\texttt{\textbackslash figref\{fig:sample\}} \xrightarrow{\text{実行時}} \texttt{\textbackslash figurename\textbackslash ref\{fig:sample\}} \xrightarrow{\text{表示例}} \text{図 1}$$

\figurename コマンドは特に設定を変更しない限りは「図」と出力されると考えて良い[*1]．

[*1] 10.2 節の方法で自由に書き換えることが可能である．

自作コマンド作成の例

▶ 相互参照

先程例示した`\figref` のように他の相互参照についても，以下のようにコマンドを作成しておくと便利である (`\figref` も再掲)：

```
\newcommand{\figref}[1]{\figurename\ref{#1}}
\newcommand{\tabref}[1]{\tablename\ref{#1}}
\newcommand{\tblref}[1]{\tablename\ref{#1}} % 略称に派閥があるらしい
\renewcommand{\equref}[1]{式 (\ref{#1})}
\newcommand{\chapref}[1]{第\ref{#1}章}
\newcommand{\secref}[1]{\ref{#1}節}
```

▶ ベクトル解析の記号

他にも，ベクトル解析の勾配・発散・回転の記号 $\mathrm{grad}, \mathrm{div}, \mathrm{rot}$ は三角関数や対数関数のように簡単なコマンドが用意されていない．そこで，以下のように作成してみた (`\div` は割り算の記号で既に存在することに注意)：

```
\newcommand{\grad}{\mathrm{grad}\,}
\newcommand{\dive}{\mathrm{div}\,}
\newcommand{\rot}{\mathrm{rot}\,}

$\grad V = -\bm{E}, \dive\bm{B} = 0,
 \rot\bm{E} = -\dfrac{\partial\bm{B}}{\partial t} $
```

$$\mathrm{grad}\, V = -\boldsymbol{E}, \mathrm{div}\, \boldsymbol{B} = 0, \mathrm{rot}\, \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t}$$

$\sin\theta$ のような既存の記号でも sin と θ の間には空白があるように，記号と変数の間は`\,` で僅かに空白を開けたほうが見た目が良くなる ($\mathrm{grad}\, V$ と $\mathrm{grad} V$ を見比べよ)．

10.2 既存のコマンドの内容を変更

既に定義されているコマンドについて，再定義することでコマンドの内容を書き換えることも可能である．新しくコマンドを作成するコマンドが`\newcommand` であったのに対し，既存のコマンドを書き換えるには`\renewcommand` コマンドを用いる．引数の指定方法については同じである．

ここでは，例として番号付き箇条書きの見た目を変える方法を紹介する．まず，デフォルトでは以下のような見た目になっている：

```
\begin{enumerate}
  \item 第1レベル
  \begin{enumerate}
    \item 第2レベル
  \end{enumerate}
\end{enumerate}
```

1. 第 1 レベル
   (a) 第 2 レベル

第 1 レベルを「1.」から「$\boxed{1}$」のように，第 2 レベルを「(a)」から「(1–a)」のように変えるには，以下のように記述する：

```
% 以下はプリアンブルに
\renewcommand{\labelenumi}{\fbox{\arabic{enumi}}}
\renewcommand{\labelenumii}{(\arabic{enumi}--\alph{enumii})}

% 以下はdocument 環境内に
\begin{enumerate}
  \item 第1レベル
  \begin{enumerate}
    \item 第2レベル
  \end{enumerate}
\end{enumerate}
```

$\boxed{1}$ 第 1 レベル
(1–a) 第 2 レベル

第 1 レベルの箇条書きの体裁は`\labelenumi` コマンドで，第 2 レベルの箇条書きの体裁は`\labelenumii` コマンドで定義されているため，これらのコマンドを書き換えれば良いわけである．どのような順番で表現するのかについては表 10.1 に示すようなコマンドを用い，引数は第 1 レベルなら `enumi`，第 2 レベルなら `enumii` のように記述する．

表 10.1　番号付き箇条書きの変更コマンド

コマンド	効果
\arabic	アラビア数字 (1, 2, 3)
\alph	小文字アルファベット (a, b, c)
\Alph	大文字アルファベット (A, B, C)
\roman	小文字ローマ数字 (i, ii, iii)
\Roman	大文字ローマ数字 (I, II, III)

なお，第 3 レベルについては\labelenumiii や enumiii，第 4 レベルについては\labelenumiv や enumiv を用いれば良い[*2]．

[*2] コマンド名の末尾がローマ数字になっている．同様に，番号無し箇条書きについては\labelitem**のように**がレベル別のローマ数字になるだけである．

付録 A

知っておくと便利なパッケージ等

ここでは，利用は必須ではないが存在を知っておくと便利なパッケージやクラスを簡単に紹介する．パッケージによっては，かなり大規模なドキュメンテーションが用意されているようなものも存在するため，ここで全てを紹介するのは困難である．

A.1 url

url パッケージは，文字通り URL を貼り付けるのに適したパッケージである．導入するにはプリアンブルに以下のコマンドを記述する：

```
\usepackage{url}
```

文書中に URL を貼り付ける際は\url コマンドを用いる：

```
https://www.example.com/

\url{https://www.example.com/}
```

---

https://www.example.com/
https://www.example.com/

これにより，URL 部分のフォントが変わるだけでなく，「URL として」認識されるようになる．つまり，PDF 表示中に URL 部分をクリックすると自動で Web ブラウザが起動され，その Web ページが表示される．

A.2 siunitx

siunitx パッケージは，物理量の単位を綺麗に表示できるパッケージである．利用するには，プリアンブルに以下を記述する：

```
\usepackage{siunitx}
```

物理量全体を記述する

物理量の値が文字でなく，数値として定まっている場合については数値と単位の両方を 1 つのコマンドで記述できる．この場合は\SI コマンドを用い，第 1 引数に数値を，第 2 引数に単位を指定する．

```
\SI{1.5}{cm}, \SI{120}{k\Omega}
```

$1.5\,\mathrm{cm}$, $120\,\mathrm{k\Omega}$

単位や SI 接頭辞を用いる場合，専用のコマンド[*1]もあるのだが，普段のアルファベットやギリシャ文字のコマンドでも同じ結果が得られるので，そちらを用いたほうがコマンドを調べたりする必要がないので楽であるだろう．

単位のみを記述する

場合によっては，数値の部分が文字になっていたりすることもある．そのような場合は，\si コマンドを用いれば単位の部分のみを記述することが可能である．第 1 引数に単位を指定する．

```
$I \si{[A]}$
```

$I[\mathrm{A}]$

このコマンドは，数式環境内部に記述しても良い (\SI も同様)．文字と単位を差別化するための括弧もコマンドの内部に記述して良い．

[*1] 例えば [mA] を表現する場合に\si{\milli\ampere}と入力することも可能である．

A.3 TikZ

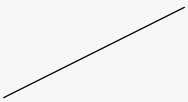
TikZ は，図形を描画するための機能を備えたパッケージである．これを利用すれば幾何図形・数学関数のグラフ・回路図などをコマンドで作成できる．ここでは簡単なコマンド例や作例の紹介のみ行う．TikZ を利用するにはプリアンブルに以下を記述する：

```
\usepackage[dvipdfmx]{graphicx}
\usepackage{tikz}
```

基本的な図形描画法

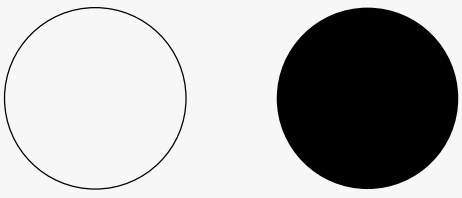
TikZ で作図を行うコマンドは tikzpicture 環境内に記述する[*2]．例えば，線分を記述する場合は\draw コマンドを利用し，座標を-- で繋げる：

```
\centering
\begin{tikzpicture}
 \draw (1, 2) -- (3, 3);
\end{tikzpicture}
```



\fill コマンドを用いると図形に色を塗ることができる：

```
\centering
\begin{tikzpicture}
  \draw (0, 0) circle [radius=1];
  \fill (3, 0) circle [radius=1];
\end{tikzpicture}
```



[*2] キャプションを利用したい場合は figure 環境内に tikzpicture 環境を入れ子で記述する．

数学関数のグラフ

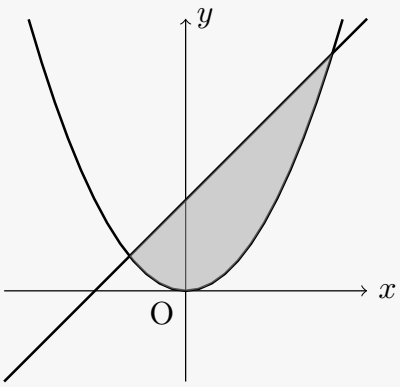
以下は，関数 $y = x^2$ と $y = x + 1$ のグラフと 2 本のグラフに囲まれた領域に色を塗った図の描画例である：

```
\centering
\begin{tikzpicture}
 % 座標軸
 \draw[->] (-2,0) -- (2,0) node[right] {$x$};
 \draw[->] (0,-1) -- (0,3) node[right] {$y$};
 \draw (0, 0) node[below left] {O};

 % y = x^2
 \draw[thick, domain=-1.73:1.73] plot (\x, {\x*\x});

 % y = x + 1
 \draw[thick, domain=-2:2] plot (\x, {\x + 1});

 % 領域の塗りつぶし
 \fill[gray!70, opacity=0.5]
   ({(1 - sqrt(5))/2}, {((1 - sqrt(5))/2)^2})
   plot[domain={(1 - sqrt(5))/2}:{(1 + sqrt(5))/2}] (\x, {\x*\x})
   -- cycle ;
\end{tikzpicture}
```



TikZ ライブラリと作図の応用例

TikZ には更に機能を拡張することが可能なライブラリが存在する．ライブラリを導入するにはプリアンブルに`\usetikzlibrary` コマンドを記述してライブラリを指定する．例えば，論理回路に関するライブラリを利用したい場合はプリアンブルに以下のコマンド

を記述する：

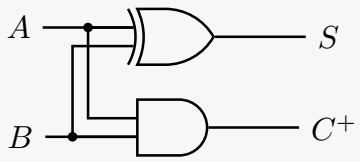
```
\usetikzlibrary{circuits.logic.US}
```

このライブラリを利用すれば以下のような論理回路 (半加算器) の描画も行える：

```
\centering
\begin{tikzpicture}[circuit logic US, thick]
 \tikzstyle{dot} = [fill, shape=circle, minimum size=3pt, inner sep=0
     pt]

 \draw (0,0) node[and gate] (and) {};
 \draw (and)+(0,1) node[xor gate] (xor) {};

 \draw (xor.output) -- ++ (1,0) node[right] {$S$};
 \draw (and.output) -- ++ (1,0) node[right] {$C^{+}$};
 \draw (xor.input 1) -- ++ (-1,0) node[left] {$A$};
 \draw (and.input 2) -- ++ (-1,0) node[left] {$B$};
 \draw (xor.input 1) -- ++ (-0.5,0) node[dot] {} |- (and.input 1);
 \draw (and.input 2) -- ++ (-0.7,0) node[dot] {} |- (xor.input 2);
\end{tikzpicture}
```



他にも，フリップフロップ回路 (図 A.1) やオペアンプ回路 (図 A.2) を作成することも可能である．なお，オペアンプ回路については `circuitikz` というライブラリを用いている．

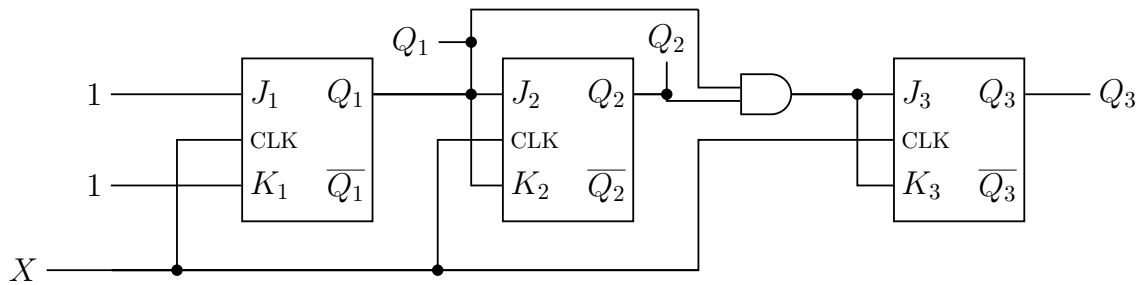


図 A.1　フリップフロップ回路の作成例

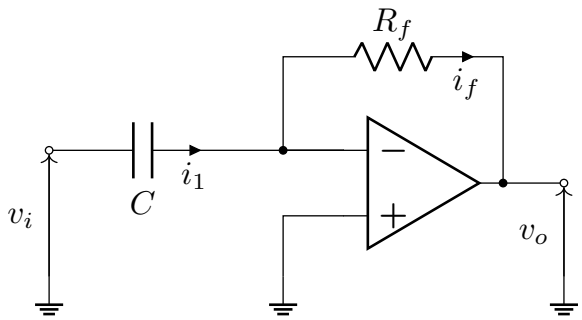


図 A.2　オペアンプ回路の作成例 (circuitikz が必要)

A.4 tcolorbox

tcolorbox は，多種多様なデザインの枠を作成できるパッケージである．本テキストのコマンドラインの表示例や記法の表示例についても，tcolorbox を用いて作成している．ここでは簡単なコマンド例の紹介のみ行う．tcolorbox を利用するにはプリアンブルに以下を記述する：

```
\usepackage{tcolorbox}
```

枠を作成するには tcolorbox 環境を利用する．枠の基本形は以下の通りである：

```
\begin{tcolorbox}
 文字が枠で囲まれる
\end{tcolorbox}
```

文字が枠で囲まれる

ここから，様々なオプションを指定することで自分独自の枠をデザインすることができる．例えば，見出しをつけたり色を変えたりすることもできる：

```
\begin{tcolorbox}[
  title=枠の見出しをつける，% 見出し
  fonttitle=\large\bfseries\sffamily，% タイトル部分のフォント指定
  fontupper=\sffamily，% 枠内のフォント
  colframe=blue!70，% 見出しの背景色
  colback=blue!5，% 枠の背景色
  arc=0pt，% 四隅の角丸部分の半径（0pt は sharp corners と同等）
  boxrule=0.5pt % 枠線の太さ
]
 枠の中身にはもちろん$x=1$や
 \[ \int x^2 \, dx = \frac{1}{3}x^3 + C \]
 のように数式環境を記述することもできる
\end{tcolorbox}
```

枠の見出しをつける

枠の中身にはもちろん $x = 1$ や

$$\int x^2 \, dx = \frac{1}{3}x^3 + C$$

のように数式環境を記述することもできる

毎回枠を作成するたびにオプションを指定するのは煩雑である．そのような場合は\newcolorbox コマンドを利用することにより自分独自の枠を呼び出す環境[*3]を作成できる．例えば，先ほど作成した枠を mybox 環境として作るには，プリアンブルに以下を記述する：

```
\newtcolorbox{mybox}[2][]{
  title={#2}，% 見出し
  fonttitle=\large\bfseries\sffamily，% タイトル部分のフォント指定
  fontupper=\sffamily，% 枠内のフォント
  colframe=blue!70，% 見出しの背景色
  colback=blue!5，% 枠の背景色
  arc=0pt，% 四隅の角丸部分の半径（0pt は sharp corners と同等）
  boxrule=0.5pt % 枠線の太さ
}
```

[*3] このテキストでは扱っていないが，環境を自作することも可能である．

これにより，mybox 環境が利用できるようになる：

```
\begin{mybox}{タイトル}
  簡単に自作の枠を呼び出せるようになった！
\end{mybox}
```

タイトル

簡単に自作の枠を呼び出せるようになった！