TEX の使い方*

久木田水生

2011 年　京都大学情報科学演習

1　はじめに

1.1　TEX とは？

TEX はアメリカの数学者，コンピュータ・サイエンティスト，Donald Knuth によって開発された，組版用のソフトウェアである．組版（植字；typesetting）とは，印刷工程の一つで，原稿とレイアウトに基づいて，文字や図などを 1 ページ毎に，印刷する形にまとめることである．以前はこの作業は手作業で行われていたのであるが，現在ではほとんどコンピュータを使って行われている．TEX は組版の作業を効率よく行うために開発されたもので，特に数式などを含む文書を簡単に美しく組版することができる．

TEX にかかわるプログラムはすべて無料で手に入れることができる．また TEX はユーザによるカスタマイズや拡張が容易であるため，自分で必要なマクロを定義したり，他のユーザの作成したマクロを利用したりすることができる．TEX ユーザの数は多く，特に数学系の論文は TEX を使って書くのが一般的になっている．現在多くの学会が，TEX に基づいて学会誌のスタイルを決定している．

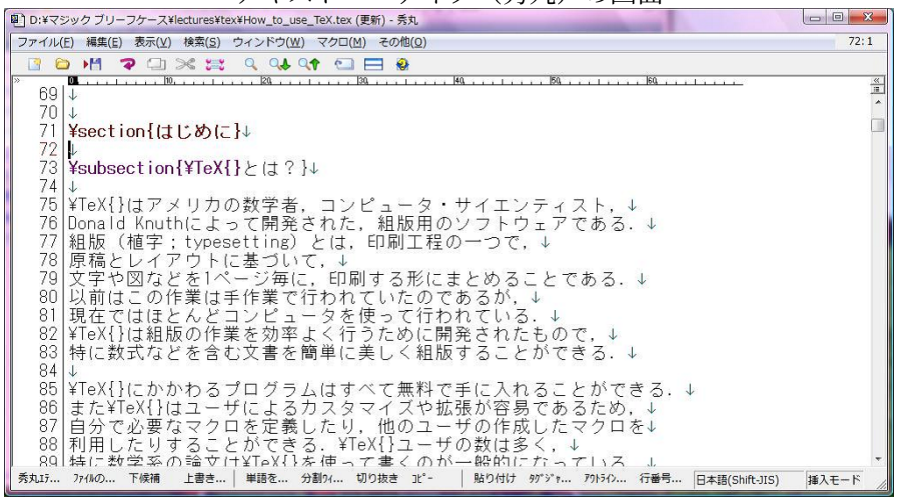
1.2　TEX とワープロソフトの違い

Microsoft の Word などに代表されるワープロソフトを使う場合，原稿を編集する作業と組版の作業とが同時に行われ，編集している画面がそのままの形で印刷されることになる．一方 TEX を使って原稿を作成する際には，編集作業はテキストエディタなどで，組版は TEX で，原稿の閲覧や印刷は Dviout や Acrobat Reader などの文書閲覧ソフトで，という形になる．このことは若干回りくどく，面倒に思われるかもしれないが，TEX での作業を効率よく行うように作られたエディタが数多く存在し，また現在のコンピュータの性能では TEX での処理にかかる時間もわずかなので，それほど作業の障害になることはない．

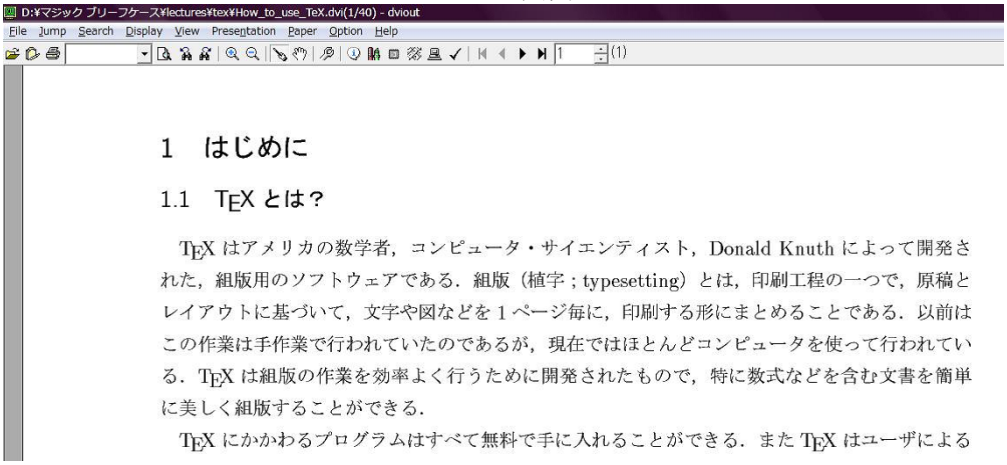
---

* Cf. 奥村晴彦『LATEX2e 美文書作成入門』，第 2 版，技術評論社，2000．海上忍，黒川弘章『これだけで出来る LATEX 実践活用ガイド』，技術評論社，2000 年．乙部厳己，江口庄英『pLATEX2e for Windows: Another Manual』, Vol. 1, 2, ソフトバンクパブリッシング，1997

テキスト・エディタ（秀丸）の画面



DVI の画面



テキストエディタは，文字情報のみを編集，保存するソフトである．フリーウェアも含め，様々な種類があるが，使っているエディタや OS の違いに関わりなくファイルのやり取りができる．この授業では，京大のメディアセンターの端末にインストールされている EasyTEX というエディタを使うことを前提とする．TEX を利用するのに便利なエディタとしては他に，LabEditor，WinShell などがある．

TEX 用の文書を作成するときは，テキストエディタで「.tex」という拡張子を持つファイルを作り，そのファイルの中に TEX に特有の命令を書き込んでおく．TEX でそのファイルを処理する（これを「TEX でコンパイルする」という）ことによって「.dvi」という拡張子を持つファイルが作られる．DVI というアプリケーションによってそのファイルを開くと，組版された文書が現れる．なお MaC の場合は DVI ファイルではなく，PDF ファイルになるらしい．Windows でも大抵の場合，簡単に DVI から PDF に変換することができる．DVI は TEX ユーザのコンピュータでなければ通常は入っていないが，PDF に変換すればほとんどのコンピュータで閲覧が可能である．

TEX は Word と違い，コンパイルしなければ文書がどのように仕上がるかがわからない，命令の仕方を間違えると思った通りに仕上がらない，といったことが起こりうる．また TEX には非常に多くの命令があり，それらを適切に使いこなせるようになるまでには，ある程度の慣れが必要である．とっつきやすさ，直感的な使い勝手，といった点では，ワープロソフトの方に利点があるだろう．一方で TEX は慣れれば慣れるほど，自由に色々なことができるという利点がある．

1.3 TEX の長所と短所

長所：

- 自動ハイフネーション，リガチャなどの組版技術が組み込まれている．
- 数式が簡単に美しく書ける．
- 章節などの構造化が容易．
- 文献，索引，目次，柱，脚注，相互参照などの機能が充実しており，細かいカスタマイズができる．
- レイアウトの微細な調整ができる．（約 0.000005mm の精度がある）
- OS に依存しない．
- 自分の好きなテキスト・エディタで編集できる．
- 自前の命令やスタイルが定義できる．

短所：

- インストールに手間がかかる．
- エラーが出てコンパイルできない時がある．
- 字体，フォント，文字サイズなどのバリエーションが少ない．
- 直感的に操作できない．
- 表やグラフなどが簡単に描けない．
- 色々なコマンドを使いこなすのが面倒．
- TEX のインストールされているコンピュータが少ない．

2 とにかく使ってみる

2.1 TEX ファイルを作る

ここではテキスト・エディタとして EasyTEX を使う場合について説明する[*1]．まず EasyTEX で新しいファイルを作る．拡張子は「.tex」とすること．

[*1] EasyTEX は数学者の中川仁氏の作ったフリーの TEX 用テキストエディタである．次の Web ページからダウンロードできる．http://www.juen.ac.jp/math/nakagawa/texguide.html （2011 年 11 月 10 日）

TEX で文書を作るために最低限必要な命令は次の三つである．なお TEX での命令は一般的に\（バックスラッシュ）から始まる．これは環境によっては¥として出てくるが，同じものとして処理される．

(1) `\documentclass[`*xxx*`]{`*yyy*`}`

すべての TEX 文書の先頭に書かれる命令．その文書がどのようなタイプの文書であるかの宣言．*yyy* の部分には特定の文書クラスの名前が入る．文書クラスの名前には次のようなものがある．

- `article`：比較的短い論文やレポートのためのクラス．
- `jarticle`：日本語の論文，レポートのためのクラス．
- `book`：一冊の本のためのクラス．
- `jbook`：日本語の本のためのクラス．

以下では `jarticle` を使う場合について説明しよう．

xxx には，オプション引数として用紙のサイズや文字のポイントを指定する文字が入る．指定しなければデフォルトの値が使われる．

(2) `\begin{document}`

ここから文書が始まるという合図．

(3) `\end{document}`

ここで文書が終わるという合図．この命令以後に書かれた内容はすべて無視される．

EasyTEX で新規文書を開き，次のように書き込んでみよう．EasyTEX では，ツールバーから命令を指定することもできる．

```
\documentclass[a4paper, 11pt]{jarticle}
\begin{document}
これは\TeX{}で作った文書です．
\end{document}
```

書き終わったら名前を付けて文書を保存する．

2.2 コンパイル

TEX ファイルを作成したら，次にコンパイルする．EasyTEX のツールバーの [コンパイル] を押す．すると同じフォルダに「.dvi」という拡張子を持つ同じ名前のファイルができている．そのファイルをダブルクリックして開いても良いが，EasyTEX のツールバーの [DVI] ボタンを押しても開くことができる．DVI ファイルを開くと

> これは TEX で作った文書です．

という文字が出ている．印刷したいときは，DVI で [File] > [Print] を開いて印刷する．

TEX のコンパイルはコマンドプロンプトを使っても出来る．コマンドプロンプトを開き，TEX ファイルが置かれているフォルダに移動し，

latex 文書名.tex

と打ち込むと TEX ファイルがコンパイルされて，dvi ファイルが作成される．このやり方であれば，TEX 用のエディタを使わなくてもコンパイルが出来る．

問題 2.1. (1) 先ほど作った文書の`\begin{document}`と`\end{document}`の間に，いろんな文字列を打ち込んで，上書き保存，コンパイルして，dviout でどのように見えるか試しなさい．
(2) 色々な記号を全角と半角の両方で入れて，違いを調べなさい．
(3) 半角スペースと改行（Enter）を連続して入力して，どう出力されるか調べなさい．

2.3 いくつかの注意するべきこと

- 半角スペースは連続して入れても，二個目以降は無視される．スペースを空けたいときは`\hspace{...pt}`という命令が使える．`...` のところに数字をいれると，その数字分のポイントが空く．なお 1 ポイントは約 0.3514mm である．ポイント以外にも `mm`，`cm`，`in`（インチ），などの単位を使うことも出来る．また `em` という単位は大体「m」という文字一個分の幅を表す．
- 改行記号は一個だけ入れても改行にならない．二個入れると改行されるが三個目以降は無視される．段落と段落の間を空けたいときは`\vspace{...pt}`という命令が使える．`\hspace` と同様に他の単位も使える．`ex` という単位は大体「x」という文字一個分の高さを表す．
- 以下の記号は TEX で特殊な用途を持っているため，そのまま入力しても，出力されない：

  `^`　`_`　`#`　`$`　`%`　`&`　`{`　`}`

  これらの記号を出力するためには，その前に`\`をつける．たとえば`\&`と入力すると&と出力される．
- バックスラッシュ（円マーク）の後ろに半角スペースを入力すると半角スペースが出力される．複数個書くと，その分だけスペースが空く．
- `%` の後は，改行文字が現れるまで（その改行文字を含めて）無視される．この機能はテキストにコメントを挿入するときに使われる．

3 文書を構造化する

3.1 章，節など

一般に文書は, 部(part), 章(chapter), 節(section), 小節(subsection), 小小節(subsubsection)といった階層構造を持っている．article クラスの文書では節以下の構造を指定する命令\section, \subsection, \subsubsection を使うことが出来る．これらの命令を使うと，自動的に節番号をつけてくれる．節の番号はたとえば定理の番号などにも連動させることが出来る．この文書の練習問題の番号も節番号と連動している．

たとえば,

```
\section{AAA}
aaa
\subsection{BBB}
bbb
\section{CCC}
ccc
\subsection{DDD}
ddd
\subsubsection{EEE}
eee
```

と入力すると

1　AAA

aaa

1.1　BBB

bbb

2　CCC

ccc

2.1　DDD

ddd

2.1.1　EEE

eee

というように出力される．なお TEX では節のタイトルだけが別のページにならないように自動的に調整されるようになっている．

その節の番号は`\thesection` という命令によって表示することが出来る．同様にその小節，小小節の番号は`\thesubsection`，`\thesubsubsection` という命令で表示することが出来る．ここでは

```
\thesection
\thesubsection
\thesubsubsection
```

と入力すると，

3
3.1
3.1.1

と出力される．

3.2 タイトル，筆者名，日付

文書のプリアンブルに例えば

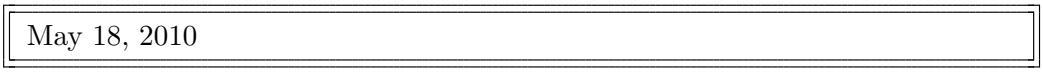
```
\title{\TeX{}の使い方}
\author{久木田水生}
\date{2010 年 5 月 18 日}
```

と書き，文書の冒頭に

```
\maketitle
```

と書くと，文書の冒頭に

のように出力される．なお\date{\today}と書いておくと，コンパイルした日の日付が出力される．jarticle クラスなどでは日本語で日付が出力されるが，article クラスなどでは英語で



などと出力される．

問題 3.1. (1) 節などの構造を持つ文書を作りなさい．
(2) 異なる節で\thesection によって節番号を参照しなさい．
(3) maketitle 命令を使って，文書にタイトル，著者名，日付をつけなさい．

3.3 節番号を操作する

節などの番号は 1 から始まり，新しい節が作られるたびに 1 ずつあがっていく．しかし例えば最初の節番号を 0 にしたいときもあるだろう．そのようなときは文書の最初に

```
\setcounter{section}{-1}
```

とつけると，最初の節が 0 から始まる．一般に第二引数を整数 n としておけば次の\section 命令で始まる節番号が $n+1$ になる．章，小節，小々節などについても同様である．

3.4 脚注

TEX には脚注[*2]をつける機能がある．脚注をつけるには，脚注をつけたい言葉の後ろに\footnote{...}と書く．「...」の部分に書いた文字が脚注としてそのページの下に印刷される．

3.5 番号の参照

TEX を使うと章節，脚注などの番号を自動的に管理させることができる．また数学の論文などを書く際には，定義や定理などの番号も TEX に管理させることができる．これらの番号を参照する必要が出てくる場合がしばしばある．各章節や脚注，定理などにラベルをつけておくと，そのラベルの名前で番号を呼び出すことができる．ラベルを付けるには

```
\label{xxx}
```

という命令を使う．*xxx* には好きな名前を付けることができるが，参照したいものと関連のある名前を付けるのが良いだろう．呼び出すときは

```
\ref{xxx}
```

という命令を使う．

たとえばこの小節の冒頭には\subsection{**番号の参照**}\label{REF} と書いてある．よって\ref{REF}によって 3.5 が表示される．なお\pageref という命令を使うと，ページ数を参照することができる．よって\pageref{REF}によって 9 が表示される．

脚注の中にラベルをおいておけば脚注の番号を参照することができる．上の脚注には

```
\footnote{脚注とはこのようなものである． \label{SampleFootnote}}
```

と書いておいたので，\ref{SampleFootnote}によって，*2 が表示される．

3.6 目次

\tableofcontents という命令で目次を出力することができる．

問題 3.2. (1) 文書に脚注をつけ，その番号を\label と\ref を使って参照しなさい．

(2) 文書に節，小節を設け，その番号を\label と\ref を使って参照しなさい．

(3) 目次を作りなさい．

[*2] 脚注とはこのようなものである．

4 環境

TEX では

```
\begin{xxx}
...
\end{xxx}
```

の形をした命令が多数存在する．この\begin{xxx}と\end{xxx}に挟まれた部分を環境と呼ぶ．ある環境の中では，その環境に独特な書式や命令が定義されている．ここではいくつかの環境を紹介しよう．

4.1 quote 環境

文書中で他の文献から文章を引用したい場合，短い文章ならば「」に入れて，通常の段落の中に入れて引用する．数行にわたる場合は，別段落にして，地の文よりも左の空欄（インデント）を多くあけて引用する．またその際，地の文章との間を，通常の段落の間よりも大きくあける．quote 環境はこのような引用のための書式を定義している．例えば次のようにテキストに書き込むと，

```
Vogt は次のように述べる．
\begin{quote}
意味は記号がどのように，そしていかなる機能とともに構成されるか ということに依存すると私は論じてきた．そのようなものとして， 記号の意味はエージェントの身体的経験，ならびにエージェントと 指示対象との相互作用に基く，形式と指示対象の間の機能的関係と 見なされうる．(Vogt, 2007, 180)
\end{quote}
```

結果は次のように出力される．

Vogt は次のように述べる．

> 意味は記号がどのように，そしていかなる機能とともに構成されるかということに依存すると私は論じてきた．そのようなものとして，記号の意味はエージェントの身体的経験，ならびにエージェントと指示対象との相互作用に基く，形式と指示対象の間の機能的関係と見なされうる．その経験はエージェントの，指示対象および／または形式との相互作用の歴史に基づく．(Vogt, 2007, 180)

なお quote 環境と似たものに，quotation 環境があるが，こちらは引用文中で段落の冒頭を字下げするようになっている[*3].

4.2 itemize 環境，enumerate 環境，description 環境

箇条書きを作る環境．例えば

```
\begin{itemize}
\item PFM
\item Banco del Mutuo Soccorso
\item Area
\end{itemize}
```

と入力すると，

- PFM
- Banco del Mutuo Soccorso
- Area

と出力される．箇条書きを入れ子にすることもできる．

```
\begin{itemize}
\item PFM
  \begin{itemize}
  \item Storia di Un Minuto
  \item Per Un Amico
  \end{itemize}
\item Banco del Mutuo Soccorso
  \begin{itemize}
  \item Io Sono Nato Libero
  \item Darwin!
  \end{itemize}
\end{itemize}
```

と入力すると次の出力が得られる．

[*3] TEX では通常，段落の先頭で字下げが行われる．字下げをしたくないときは，その段落の前に\noindent という命令を置く．

- PFM
  - Storia di Un Minuto
  - Per Un Amico
- Banco del Mutuo Soccorso
  - Io Sono Nato Libero
  - Darwin!

enumerate 環境を使うと番号付きの箇条書きができる．

```
\begin{enumerate}
\item PFM
\item Banco del Mutuo Soccorso
\item Area
\end{enumerate}
```

と入力すると，

1. PFM
2. Banco del Mutuo Soccorso
3. Area

と出力される．

description 環境は，指定した文字列を使って箇条書きを作る環境である．\item の後ろに [] でくくった文字列をつける．例えば

```
\begin{description}
\item[日時] 5 月 25 日 16 時 30 分から
\item[場所] 京大メディアセンター南館 204
\item[内容] \TeX{}の使い方
\end{enumerate}
```

と入力すると次の出力が得られる．

日時　5 月 25 日 16 時 30 分から
場所　京大メディアセンター南館 204
内容　TEX の使い方

4.3 center 環境，flushright 環境，flushleft 環境

それぞれ文字を中央寄せ，右寄せ，左寄せする．例えば

```
\begin{flushleft}
受講生各位
\end{flushleft}
\begin{flushright}
情報科学演習担当

久木田水生
\end{flushright}
\begin{center}
レポートについて
\end{center}
```

という入力に対して

受講生各位

情報科学演習担当
久木田水生

レポートについて

という出力が得られる．

4.4 tabular 環境

表を作る環境．たとえば次のように書く．

```
\begin{tabular}{|r|c||l|}
\hline
AAAA & BBB & CCC \\ \hline\hline
D & E & F \\ \hline
G & H & I \\ \cline{2-3}
 & J & K \\ \hline
\end{tabular}
```

すると次の出力が得られる．

AAAA	BBB	CCC
D	E	F
G	H	I
	J	K

`\begin{tabular}`の後の`{}`の中には `c`，`l`，`r`，`|`[*4]が入る．文字の数は作られるカラムの数，`c`，`l`，`r` はそれぞれ，そのカラムで中央寄せ，左寄せ，右寄せにすることを指定する．

文字の間の`|`の有無は，カラムの間の区切りの罫線の有無を指定する．上の例では，カラムは 3 列，左から右寄せ，中央寄せ，左寄せにすること，全てのカラムの両端に罫線，特に 2 列目と 3 列目の間には二重罫線を入れることを指定している．

カラムの区切りは`&`によって，行の区切りは`\\`によって指定する．行と行の間に`\hline` という命令をおくと水平の罫線が引かれる．また`\cline` は指定したカラムの分だけ水平の罫線を引く命令である．

TEX では一つの表は一つのオブジェクトとして扱われるので，複数のページにまたがる表を作ることはできない．複数のページにまたがる表を作るための命令も作られているが，ここでは説明しない．基本的に TEX はそのような大きな表を作ることには向いていないと思った方が良い．

表を通常の段落の中に配置することもできる．たとえば

A	B
C	D

のように．

表の中に表を埋め込むこともできる．例えば

a	b	e	f
c	d		
g		h	i

のように．

`|`や`\hline` を重ねることで罫線を二重線，三重線，・・・にすることができる．たとえば次のように．

A	B
C	D

複数のカラムをまとめるときは`\multicolumn{`x`}{`y`}{...}` という命令を使う．x には数字が入り，何個のカラムをまとめるかを指定する．y には `c`，`l`，`r` のいずれかが入り，中央寄せ，左寄

[*4] [Shift] + `¥`.

せ，右寄せを指定する．たとえば次の入力に対して

```
\begin{tabular}{|l||c|r|}
\hline
\multicolumn{3}{|c|}{AAAAAAAAAAAAAAAAAA} \\ \hline\hline
\multicolumn{2}{|r|}{BBB} & CCC \\ \hline
D & \multicolumn{2}{|l|}{EEEE} \\ \hline
F & GGGGGG & \multicolumn{1}{|c|}{H} \\ \hline
\end{tabular}
```

次の出力が得られる．

AAAAAAAAAAAAAAAAAA		
	BBB	CCC
D	EEEE	
F	GGGGGG	H

問題 4.1. tabular 環境を使って以下の表を作りなさい．

(1)

9								
	8							
		7						
			6					
				5				
					4			
						3		
							2	
								1

(2)

AAAAAAAAAAAAAAAAAA		
BBB	CCC	
	DD	EE

5 字体とサイズ

TEX では文字のサイズと字体を指定することができる．
字体を指定する命令には次のようなものがある．

- \textgt{abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五}
  $\Longrightarrow$ abcdeFGHIJ01234 **あいうえおカキクケコ壱弐参四五**
- \textbf{abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五}
  $\Longrightarrow$ **abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五**
- \textit{abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五}
  $\Longrightarrow$ *abcdeFGHIJ01234* あいうえおカキクケコ壱弐参四五
- \textsf{abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五}
  $\Longrightarrow$ abcdeFGHIJ01234 **あいうえおカキクケコ壱弐参四五**
- \texttt{abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五}
  $\Longrightarrow$ abcdeFGHIJ01234 **あいうえおカキクケコ壱弐参四五**
- \textsl{abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五}
  $\Longrightarrow$ *abcdeFGHIJ01234* あいうえおカキクケコ壱弐参四五
- \textrm{abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五}
  $\Longrightarrow$ abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五
- \textsc{abcdeFGHIJ01234 あいうえおカキクケコ壱弐参四五}
  $\Longrightarrow$ ABCDEFGHIJ01234 あいうえおカキクケコ壱弐参四五

サイズを変える命令には以下のものがある．

- {\tiny あいうえお} $\Longrightarrow$ あいうえお
- {\scriptsize あいうえお} $\Longrightarrow$ あいうえお
- {\footnotesize あいうえお} $\Longrightarrow$ あいうえお
- {\small あいうえお} $\Longrightarrow$ あいうえお
- {\normalsize あいうえお} $\Longrightarrow$ あいうえお
- {\large あいうえお} $\Longrightarrow$ あいうえお
- {\Large あいうえお} $\Longrightarrow$ あいうえお
- {\LARGE あいうえお} $\Longrightarrow$ あいうえお
- {\huge あいうえお} $\Longrightarrow$ あいうえお
- {\Huge あいうえお} $\Longrightarrow$ あいうえお
- {\HUGE あいうえお} $\Longrightarrow$ あいうえお

字体やサイズに関しては TEX はそれほどバラエティに富んでいるわけではない．張り紙やビラのようなものを作るなら Word や PowerPoint などの方が向いているだろう．

6　アクセント記号，特殊なアルファベット

ä, á などのアクセント記号や，ßやæのような特殊なアルファベットは次のように入力して得られる．

入力	出力	入力	出力	入力	出力
\"{a}	ä	\'{a}	á	\`{a}	à
\^{a}	â	\c{a}	a̧	\~{a}	ã
\={a}	ā	\.{a}	ȧ	\u{a}	ă
\v{a}	ǎ	\H{a}	a̋	\t{a}	a͡a
\d{a}	ạ	\b{a}	a̲	\r{a}	å
\ss	ß	\ae	æ	\oe	œ
\AE	Æ	\OE	Œ		

7　数式，数学記号

7.1　数式環境

数式や数学記号は数式環境の中で使われる．数式環境を作る方法はいくつかある．$(xy)z = x(yz)$ のように，段落の中に数式を使うときは`$(xy)z = x(yz)$`のように，二つの`$`で式を挟む．

式を独立した行（別行立て）にするときは`\[`と`\]` で式を挟む．例えば

```
結合律とは
\[
(xy)z = x(yz)
\]
という規則である．
```

と入力すると

結合律とは

$$(xy)z = x(yz)$$

という規則である．

のように出力される．

数式に番号をつけたいときは `equation` 環境を使う．すると，

結合律とは

$$(xy)z = x(yz) \tag{1}$$

という規則である．

のように数式に自動的に番号が付される．節番号などと同じように数式番号も\label と\ref を使って参照することができる．上の式には

```
\begin{equation}\label{assoc}
(xy)z = x(yz)
\end{equation}
```

とラベルを付けておいたので\ref{assoc}によって 1 が出力される．数式につけられる番号は equation という名前のカウンタによって管理されている．従って例えば \setcounter{equation}{100} によって，この命令の次の数式番号は 101 になる．

数式環境について注意することがいくつかある．

- 数式環境中では通常のアルファベットは斜体になる．数式環境中でアルファベットや数字の字体を変えるには次のような命令がある．

  \mathrm{abcABC012} $\Longrightarrow$ $\mathrm{abcABC012}$

  \mathit{abcABC012} $\Longrightarrow$ $\mathit{abcABC012}$

  \mathbf{abcABC012} $\Longrightarrow$ $\mathbf{abcABC012}$
- 数式環境の中では半角スペースは基本的に無視される．従って$x + y$と書いても$x+y$と書いても結果は変わらない．
- 数式環境中で日本語を使いたいときは\mbox{...}によって出す．たとえば

  ```
  $\mbox{任意の}x\mbox{に対して}f(x) = g(x)$
  ```

  という入力に対しては次の出力が得られる．

  任意の x に対して $f(x) = g(x)$

7.2 数式環境の中で使える特殊な文字，記号（引数なし）

入力	出力	入力	出力	入力	出力
\infty	∞	\emptyset	$\emptyset$	\aleph	$\aleph$
\top	$\top$	\bot	$\bot$	\angle	$\angle$
\triangle	$\triangle$	\diamond	$\diamond$	\diamondsuit	$\diamondsuit$
\spadesuit	$\spadesuit$	\clubsuit	$\clubsuit$	\heartsuit	$\heartsuit$
\sharp	$\sharp$	\natural	$\natural$	\flat	$\flat$
\dagger	$\dagger$	\ddager	$\ddagger$		

7.3 関数

数式環境では様々な関数記号，関数表現が使える．

入力	出力	入力	出力
`\frac{a}{b}`	$\frac{a}{b}$	`\sqrt[a]{b}`	$\sqrt[a]{b}$
`a^{b}`	a^b	`a_{b}`	a_b
`a^{b}_{c}`	a^b_c	`|a|`	$\|a\|$
`\sin x`	$\sin x$	`\cos x`	$\cos x$
`\tan x`	$\tan x$	`\log_{y} x`	$\log_y x$
`\int_{a}^{b} f(x)dx`	$\int_a^b f(x)dx$	`\sum_{k = 0}^{n} a_{k}`	$\sum_{k=0}^{n} a_k$
`\prod_{k = 0}^{n} A_{k}`	$\prod_{k=0}^{n} A_k$	`\coprod_{k = 0}^{n} A_{k}`	$\coprod_{k=0}^{n} A_k$
`\lim_{x \to 0} f(x)`	$\lim_{x\to 0} f(x)$	`\neg A`	$\neg A$
`\forall x A(x)`	$\forall x A(x)$	`\exists x A(x)`	$\exists x A(x)$

`\lim` や`\int` などの命令は別行立てにすると添え字の位置や記号の大きさが以下のように変わる．

$$\frac{a}{b} \qquad \lim_{x\to 0} f(x) \qquad \int_a^b f(x)dx \qquad \sum_{k=0}^{\infty} a_k$$

なお段落中にも別行立ての体裁で出したいときには

```
$\displaystyle{\lim_{x \to \infty}f(x)}$
```

と段落中に書けば $\lim_{x\to\infty} f(x)$ のように出力される．

`\sqrt[a]{b}`の `[a]` はオプションで，これを書かずに`\sqrt{b}`とだけ書くと $\sqrt{b}$ と出力される．

問題 7.1. 以下の式を書きなさい．

(1) $\dfrac{\sqrt{2}}{3}$

(2) $x^y x^z = x^{y+z}$

(3) $F(x) = \displaystyle\int^x f(t)dt$

(4) $a_{n+1} = \displaystyle\sum_{k=0}^{n} 2^{a_k}$

(5) $\displaystyle\lim_{x\to 0} \frac{x^2 - 2x + 1}{x - 1}$

7.4 二項演算子

入力	出力	入力	出力	入力	出力
\cdot	$\cdot$	*	$*$	+	$+$
-	$-$	\pm	$\pm$	\mp	$\mp$
\div	$\div$	\times	$\times$	\bullet	$\bullet$
\circ	$\circ$	\ast	$\ast$	\star	$\star$
\cup	$\cup$	\cap	$\cap$	\setminus	$\setminus$
\vee	$\vee$	\wedge	$\wedge$	\otimes	$\otimes$
\oplus	$\oplus$	\ominus	$\ominus$	\odot	$\odot$

$\cup$ などには\bigcup などの大きな字体が用意されていて，これは別行立てにすると，$\sum$ と同じような添え字をつけることができる．たとえば\[\bigcup_{k = 0}^{\infty} A_{k} \] という入力に対しては

$$\bigcup_{k=0}^{\infty} A_k$$

と出力される．\cap, \vee, \wedge, \otimes, \oplus についても同様.

7.5 関係子

入力	出力	入力	出力	入力	出力
<	$<$	>	$>$	\le	$\le$
\ge	$\ge$	\prec	$\prec$	\preceq	$\preceq$
\succ	$\succ$	\succeq	$\succeq$	\subset	$\subset$
\subseteq	$\subseteq$	\supset	$\supset$	\supseteq	$\supseteq$
\in	$\in$	\notin	$\notin$	\ni	$\ni$
\vdash	$\vdash$	\dashv	$\dashv$	\models	$\models$

入力	出力	入力	出力	入力	出力
=	$=$	\neq	$\neq$	\propto	$\propto$
\equiv	$\equiv$	\sim	$\sim$	\simeq	$\simeq$
\approx	$\approx$	\cong	$\cong$	\subset	$\subset$
\smile	$\smile$	\frown	$\frown$	\mid	$\mid$
\parallel	$\parallel$	\bowtie	$\bowtie$	\perp	$\perp$

関係子を否定するには\not が使える．例えば\not \sim と書くと $\not\sim$ と出る．

7.6 ギリシャ文字

ギリシャ文字の小文字は以下の命令で出力できる．

```
\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta, \theta, \iota, \kappa,
\lambda, \mu, \nu, \xi, o, \pi, \rho, \sigma, \tau, \upsilon, \phi, \chi, \psi,
\omega
```

ϵ（epsilon）などには異なる字体もある．`\varepsilon` と書くと ε と出力される．他に異なる字体のあるものは pi, sigma, theta, rho, phi である．

ギリシャ文字の大文字は以下のものが用意されている．

```
\Gamma, \Delta, \Theta, \Lambda, \Xi, \Pi, \Sigma, \Upsilon, \Phi, \Psi, \Omega
```

問題 7.2. ギリシャ文字を出力してみなさい．

7.7 括弧，区切り記号，矢印など

入力	出力	入力	出力	入力	出力
`(x)`	(x)	`\{ x \}`	$\{x\}$	`[x]`	$[x]$
`\lceil x \rceil`	$\lceil x \rceil$	`\lfloor x \rfloorl`	$\lfloor x \rfloor$	`\langle x \rangle`	$\langle x \rangle$

以下は左右の区別のない区切り記号である．

入力	出力	入力	出力	入力	出力
`|`	$\|$	`/`	$/$	`\\|`	$\|\|$
`\uparrow`	$\uparrow$	`\Uparrow`	$\Uparrow$	`\downarrow`	$\downarrow$
`\Downarrow`	$\Downarrow$	`\updownarrow`	$\updownarrow$	`\Updownarrow`	$\Updownarrow$

括弧類を大きくするには`\bigl`，`\bigr`, `\bigm` をそれぞれ左括弧，右括弧，区別のない区切り記号の前に置く．もっと大きくするときは `Bigl`，`biggl`，`Biggl`，`Bigr`，`biggr`，`Biggr`，`Bigm`，`biggm`，`Biggm` を置く．

`\left`，`\right` を使うと括弧の中身に応じた大きさの括弧が自動的に選ばれる．これらは範囲を明確にするために，左右両方を同時に使わなければならない．片方だけに括弧を表示したいときは，表示させない方に，括弧の代わりに．をつける．たとえば `\[\left\angle \sum_{k \ge 0} a_n \right.\]` という入力に対しては

$$\left\langle \sum_{k \ge 0} a_n \right.$$

という出力が得られる．

矢印には次のようなものがある．

入力	出力	入力	出力	入力	出力
\to, \rightarrow	$\to$	\leftarrow	$\leftarrow$	\leftrightarrow	$\leftrightarrow$
\Rightarrow	$\Rightarrow$	\Leftarrow	$\Leftarrow$	\Leftrightarrow	$\Leftrightarrow$
\longrightarrow	$\longrightarrow$	\longleftarrow	$\longleftarrow$	\longleftrightarrow	$\longleftrightarrow$
\Longrightarrow	$\Longrightarrow$	\Longleftarrow	$\Longleftarrow$	\Longleftrightarrow	$\Longleftrightarrow$
\rightharpoonup	$\rightharpoonup$	\leftharpoonup	$\leftharpoonup$	\rightharpoondown	$\rightharpoondown$
\leftharpoondown	$\leftharpoondown$	\hookrightarrow	$\hookrightarrow$	\hookleftarrow	$\hookleftarrow$

問題 7.3. 以下の式を書きなさい.

(1) $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

(2) $A \wedge \left(\bigvee_{\lambda \in \Lambda} B_\lambda \right) = \bigvee_{\lambda \in \Lambda} (A \wedge B_\lambda)$

(3) $\langle x_1, x_2 \rangle \leq \langle y_1, y_2 \rangle \Longleftrightarrow x_i \leq_i y_i (i = 1, 2)$

(4) $\forall \varepsilon \in \mathbf{R} \exists \delta \in \mathbf{R} \forall y \in \mathbf{R}. |x - y| < \delta \Longrightarrow |f(x) - f(y)| < \varepsilon$

8 パッケージの読み込み

例えば`$\square$`という命令は，通常では無定義な命令としてエラーを引き起こす．しかし`\documentclass{...}`と`\begin{document}`の間[*5] に

```
\usepackage{amssymb}
```

と書いておくと上の命令によって □ という出力が得られる．これは TEX に，`\square` という命令の定義を含む `amssymb` という定義集が収められているからである．

このテキストではこれまで単に TEX という名前を使ってきたが，厳密に言うといま私たちが使っているのは，pLATEX 2_ε という，TEX の核となるプログラムに，一定のパッケージを付け加えたものである．pLATEX 2_ε には，`amssymb` 以外にも多くのパッケージが収められている．例えば `hhline` というパッケージを読み込むと `tabular` での二重罫線の重なり方を調整することができるようになる．例えば

AAA	BBB	CCC	DDD	EEE	FFF
0	1	2	3	4	5

のように．この表は次の命令で作っている．

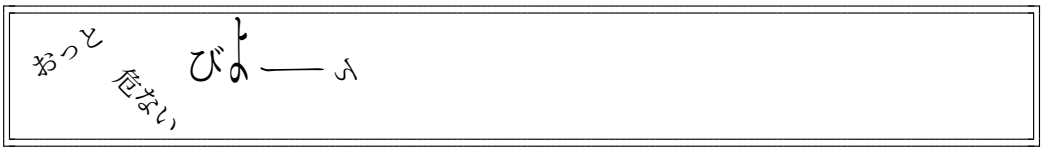
[*5] この部分をプリアンブルという．

```
\begin{tabular}[b]{||c||c||c||c||c||c||}
\hhline{|t:=t::=t::=t::=t::=t::=t:|}
AAA & BBB & CCC & DDD & EEE & FFF \\
\hhline{|:=::=||==:|=|:=:|}
0 & 1 & 2 & 3 & 4 & 5 \\
\hhline{|b:=b::=b::=b::=b::=b::=b:|}
\end{tabular}
```

また graphics というパッケージを読み込むと，文字の回転，反転，拡大縮小，縦横比変換などが出来る．例えば

```
\rotatebox{30}{おっと}\rotatebox{-45}{危ない}\scalebox{1.5}{び}
\scalebox{1}[3]{よ}\scalebox{3}[1]{ー}\reflectbox{ん}
```

によって



が出力される．

pLaTeX 2_ε には他にも様々なパッケージが収められている．pLaTeX 2_ε に収められていないパッケージでも，誰かが作ったパッケージがインターネットなどで手に入る．その場合はダウンロードしたファイルを，それを利用したい TeX ファイルと同じフォルダに入れておくと，\usepackage 命令で読み込むことができる．なお自前のパッケージの作り方については 11.5 節を参照.

9 画像の挿入

9.1 画像ファイルの取り込み

graphics を拡張した graphicx を読み込むと，上記の graphics の機能に加えて，文書の中に画像ファイルを取り込んで DVI ファイルや PDF ファイルで表示できるようになる．なお表示できる画像ファイルの種類は出力ソフトや OS などの環境に依存するが，PS ファイル，EPS ファイルならばほとんどの環境で問題なく使用できる．PS ファイル，EPS ファイルについては，例えば奥村晴彦『LaTeX2e 美文書作成入門』（第 4 版，技術評論社）を参照されたい.

graphicx パッケージを読み込むために，まずプリアンブルに

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

と書いておく．挿入したい画像ファイルを TEX ファイルと同じフォルダに置く．例えばそのファイル名が sample_picture.ps だとする．画像を表示させたい箇所に

```
\includegraphics{sample_picture.ps}
```

と書く．するとその箇所に画像が表示される．

画像の大きさは，オプション引数としてたとえば

```
\includegraphics[width=5cm]{sample_picture.ps}
```

と書くことで，指定することが出来る．この命令では画像を幅 15cm の大きさに合わせる．width の代わりに height を使うと，高さを指定することができる．両方を同時に指定することも出来る．\vspace などと同様に単位は pt，cm，mm なども使うことが出来る．

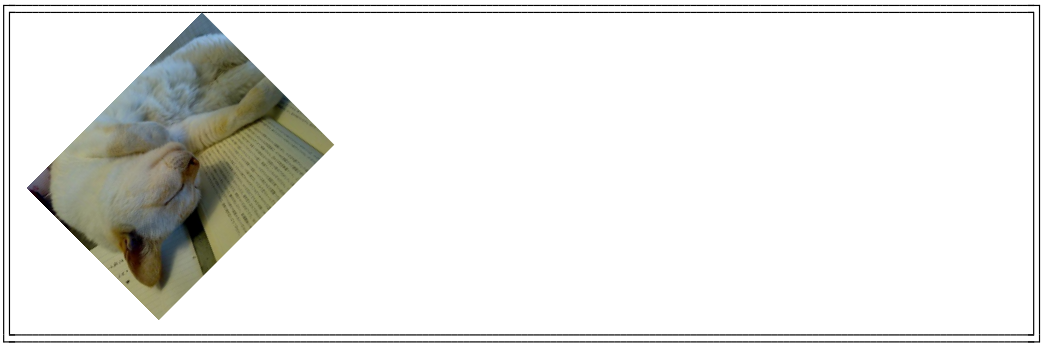
例えば

```
\includegraphics[width=3cm]{sample_picture.ps}
\includegraphics[height=3cm]{sample_picture.ps}
\includegraphics[width=6cm, height=3cm]{sample_picture.ps}
\includegraphics[width=5cm, height=3cm, keepaspectratio]{sample_p
icture.ps}
```

と入力すると，

という出力が得られる．keepaspectratio を書くと，縦横の比を保って，幅と高さの指定に収まる最大のサイズにする．

前節の\rotatebox，\scalebox，\reflectbox を画像に対して使うことも出来る．たとえば\rotatebox{45}{\reflectbox{\includegraphics[width=3cm]{sample_picture.ps}}}という命令によって



という出力が得られる．

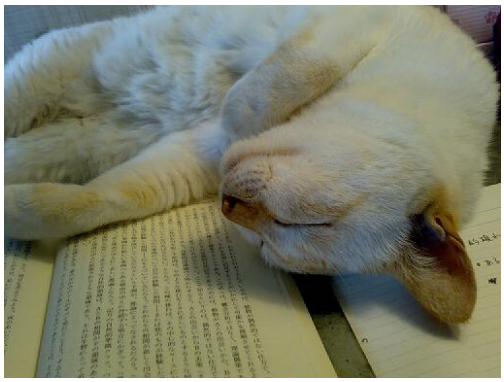


図 1　横たわる猫

9.2　画像を挿入する位置

文書中に図を挿入する際には，ページの上部や下部に置くのが普通である．figure 環境を使うと，このような図の位置の調整を TEX に任せることができる．例えば

```
\begin{figure}[t]
\begin{center}
\includegraphics[width=6cm]{sample_picture.ps}
\end{center}
\caption{横たわる猫}\label{lying_cat}
\end{figure}
```

という入力をした結果がこの頁の上部に出力されている図である．\caption 命令は図の番号と引数に与えた説明を表示させる．図の番号は TEX が自動的に管理してくれる．\caption の後ろに \label を付けることで図の番号を参照することが出来る．上では lying_cat という文字列でラベル付けをしているので\ref{lying_cat}と書くことで 1 が表示される．

\begin{figure}の後の [t] はこの図を頁の上部に出力させるための命令である．t の代わりに b，h，p を使うと，それぞれ頁の下部，その命令が書かれた場所，独立した頁に図を出力する．これらを組み合わせて使うことも出来る．例えば [htb] と書くと，まずその場所に図を挿入することを試み，それが出来ないならば上部，それが出来ないなら下部に挿入する．優先順位は h，t，b，p となっていて，書かれる順番は関係ない．従って [bht] と書いても結果は変わらない．

10　カウンタ

TEX では整数を扱うカウンタという仕組みが用意されている．章節，脚注，等式，定理などの番号はカウンタを用いて制御されている..

カウンタに関する命令には以下のようなものがある．

- \newcount：新しいカウンタを導入する命令．例えば\newcount\xxx によって\xxx という名前の新しいカウンタが導入される．カウンタの初期値は 0 になっている．
- \the\xxx：カウンタ\xxx の値を表示する．
- \xxx = n：カウンタ xxx の値を整数 n にする．
- \advance\xxx by n：カウンタ\xxx の値を整数 n だけ進めることを命じる．n は負の整数でもよい．また n には整数の代わりにカウンタ名を入れることも可能．同様の命令に \multiply，\divide がある[*6]．

なお何らかの環境中でカウンタの値を変更しても，その操作は環境の外には影響を及ぼさないことに注意しよう．たとえば

```
\newcount\xxx
\begin{quote}
\advance\xxx by 100
\the\xxx
\end{quote}
\the\xxx
```

と書くと

> 100
>
> 0

と表示される．quote 環境の中で\xxx に 100 加えられ，同じ環境の中でこのカウンタが参照されているので 100 が表示される．しかしこの環境の外で\xxx を参照すると，100 は加えられないまま（したがって初期値 0 のまま）である．

環境中での変更を環境の外でも有効にしたい場合には，変更を命じる命令の前に\global という命令を付け加える．たとえば

```
\newcount\yyy
\begin{quote}
\global\advance\yyy by 100
\the\yyy
\end{quote}
\the\yyy
```

によって

[*6] \divide は小数点以下を切り捨てる．

```
    100
100
```

という出力が得られる．

11 命令の定義

ここまでに様々な TEX の命令を紹介したが，TEX の最大の利点の一つは自前の命令を定義できることである．

TEX の命令には大きく分けて以下の三種類がある．

- 引数を取らない命令：\TeX, \alpha, \int, \noindent, etc. これらは常に一定の値を出力する．
- 引数を取る命令：\vspace, \large, \textbf, \frac, etc. これらは引数に応じた値を返す．
- 環境命令：\begin{...} \end{...}．引数に応じた値を返す．文章など，比較的長い引数をとる場合に環境を使うことが多い．

11.1 引数を取らない命令の定義

引数を取らない命令は，よく使う表現を短縮して登録しておくのに便利である．たとえば

```
\def\MyAddress{
\begin{flushright}
606-****\hspace{7em}

京都市左京区**********

久木田水生
\end{flushright}
}
```

という定義をしておいたとする．すると\MyAddress と書くだけで

606-****
京都市左京区**********
久木田水生

と出力される．

同じことは\newcommand を使ってもできる．これを使うときは

```
\newcommand{\MyAddress}{...}
```

と書く．\newcommand の場合は TEX に同じ名前の命令が既に存在するときは，コンパイルするときにエラーになる．\def の場合は元の命令を書き換えることになる．

命令の名前の中には数字および TEX で特殊な意味を持つ文字（&，#など）を使うことはできない．

問題 11.1. (1) 自前の命令，\MyEmailAddress を定義しなさい[*7]．

(2) \alpha という名前の命令を\def を使って定義した場合と\newcommand を使って定義した場合でどうなるか比較しなさい．

11.2 引数を取る命令の定義

例えば文書中の見出しとして，次のようなスタイル使いたいとする．

> **¶ モンテカルロ法**
> モンテカルロ法とは，数学者，計算機科学者，物理学者ジョン・フォン・ノイマンによって考案された，乱数を用いてある値の近似値を計算する方法である．

これは次に入力によって得られている．

```
\vspace{1ex}
\noindent
{\large \P \hspace{.5em} \textbf{モンテカルロ法}}
モンテカルロ法とは，数学者，計算機科学者，物理学者ジョン・フォン・ノイマン
によって考案された，乱数を用いてある値の近似値を計算する方法である．
```

この見出しを作るために行われていることは以下の通りである．

1. 前の段落から行を空ける：\vspace{1ex}
2. インデントさせない：\noindent
3. サイズを large にする：{\large ・・・}
4. 先頭に¶を付ける：\P
5. ¶の後ろにスペースを空ける：\hspace{.5em}
6. フォントをボールドにする：\textbf{・・・}

[*7] もしも電子メールアドレスの中にアンダーバー（_）が使われている時は注意が必要である．2.3 節を参照せよ．

7. 見出しを書く：**モンテカルロ法**

この作業において，1 から 6 まではすべての見出しに共通している．7 だけが各見出しで異なっている．このような場合，引数付きの命令として 1 から 6 の作業を定義しておくと便利である．この見出しスタイルを使いたい文書のプリアンブル[*8]に次のように書いておく．

```
\newcommand{\Heading}[1]{
\vspace{1ex}
\noindent
{\large \P \hspace{.5em} \textbf{#1}}
}
```

そうすると，この文書中で\Heading{*xxx*}（*xxx* は任意の文字列）と書けば，同じスタイルの見出しを出力することが出来る．

よく使用するスタイルを定義して登録しておくことは，作業を効率化するだけでなく，タイプミスによる間違いを減らし，かつ後々の変更を容易にする．例えば見出しのサイズを本文と同じにしたくなったとき，定義さえ書き換えればこの見出しを使っているすべての箇所にその変更が反映される．

引数つきの命令の形式は次のようなものである．

\newcommand{\xxx}[n]{zzz}

\xxx の部分には命令の名前，n の部分には引数の個数（最大で 9 個），zzz の部分には命令の内容が入る．zzz の中で引数は#1，#2，・・・，#n で表わされる．命令を使うときには

\xxx{引数 1}{引数 2}...{引数 n}

と書く．すると zzz の中に現れる#1，#2，・・・にそれぞれ引数 1，引数 2，・・・が代入された結果がそこに書かれたのと同じことになる．つまり

$$\texttt{\textbackslash xxx\{a1\}\{a2\}...\{an\}} = \texttt{zzz}[\texttt{\#1:=a1}][\texttt{\#2:=a2}]...[\texttt{\#n:=an}]\text{—}$$

が成り立つ．ただし [#k:=ak] はパラメータ#k に引数 ak を代入する操作を表す．

問題 11.2. 以下の命令を定義しなさい．

(1) 引数を 1 個受け取り，それを中央寄せ[*9]にして「***」ではさむ命令\CentAstHeading．例えば\CentAstHeading{**お知らせ**}は

[*8] 実際には，命令の定義は必ずしもプリアンブルに書かなくても，その文書中，その命令が使われるよりも前の部分に書いておけばよい．

[*9] 4.3 節を参照せよ．

お知らせ

を出力する．

(2) 1 行目に右寄せしたその日の日付，2 行目に左寄せした文書の宛先，3 行目に右寄せした自分の肩書名前等，4 行目に中央寄せして太字にした表題を出力する命令\MyHeading．表題と宛先と肩書を引数として取る．例えば\MyHeading{**レポートについて**}{**受講生各位**}{**情報科学演習担当**} は

2011 年 11 月 10 日

受講生各位

情報科学演習担当
久木田水生

レポートについて

を出力する．

(3) 引数を 1 個とり，引数をイタリック体にして，アングル ⟨ ⟩ で囲んだ文字列を出力する命令\AngIt．たとえば\AngIt{argument}に対しては $\langle argument \rangle$ が出力される．

(4) 引数を 2 個とり，$\{\#1_1, \#1_2, \ldots, \#1_{\#2}\}$ のように出力する命令\SSeq．たとえば\SSeq{x}{n}に対しては $\{x_1, x_2, \ldots, x_n\}$ が出力される．

(5) 第 1 引数にカウンタを，第 2 引数に文字列を取り，カウンタに 1 を加えた上で

(節番号.#1) #2

と出力する命令\Reibun．たとえば

```
\newcount\JimiHen
\Reibun{\JimiHen}{Let me stand next to your fire.}

\Reibun{\JimiHen}{Excuse me while I kiss the sky.}
```

という入力に対して

(11.1) Let me stand next to your fire.
(11.2) Excuse me while I kiss the sky.

が出力される．

(6) 2 引数をとり，一方をラベルとする脚注を作る命令\Lfootnote.

11.2.1 例

次のような表をよく使うとしよう.

	浦和	**鹿島**
前半	1	0
後半	2	1
合計	3	1

この表の

前半		
後半		
合計		

の部分は何度も使われる共通のパターンであり，空欄に入る文字が変化する部分である．この共通のパターンを使うような命令を次のように定義しておく.

```
\newcommand{\ScoreBoard}[8]{
\begin{tabular}{|r|c|c|}
\hline
& \textbf{#1} & \textbf{#2} \\ \hline
前半 & #3 & #4 \\ \hline
後半 & #5 & #6 \\ \hline
合計 & #7 & #8 \\ \hline
\end{tabular}
}
```

このとき

```
\ScoreBoard{浦和}{鹿島}{1}{0}{2}{1}{3}{1}
```

という入力によって

	浦和	鹿島
前半	1	0
後半	2	1
合計	3	1

という出力が得られる．

上の\ScoreBoard を改良して，合計点を自動的に計算してくれるような定義にすることも可能である．これにはカウンタを使う．例えば以下のように定義したとしよう．

```
\newcommand{\ScoreBoard}[6]{
\newcount\homesum % ホームチームの得点の合計．初期値は 0．
\newcount\awaysum % アウェイチームの得点の合計．初期値は 0．
\advance\homesum by #3 % ホームチームの前半の得点を加算．
\advance\homesum by #5 % ホームチームの後半の得点を加算．
\advance\awaysum by #4 % アウェイチームの前半の得点を加算．
\advance\awaysum by #6 % アウェイチームの後半の得点を加算．
\begin{tabular}{|r|c|c|}
\hline
& \textbf{#1} & \textbf{#2} \\ \hline
前半 & #3 & #4 \\ \hline
後半 & #5 & #6 \\ \hline
合計 & \the\homesum & \the\awaysum \\ \hline
\end{tabular}
}
```

そうすると例えば

```
\ScoreBoard{川崎}{広島}{1}{1}{2}{1}
```

という入力によって

	川崎	広島
前半	1	1
後半	2	1
合計	3	2

という出力が得られる．

上の定義では\homesum と\awaysum という二つのカウンタ（整数を表す名前）が使われている．

TEX には条件分岐やループなどの制御命令を使うことができる ifthen パッケージがあり，それによってより高度な命令を作ることもできるが，ここではその説明は省略する．

前節でも述べたように，\newcommand はすでに使われている命令と同じ名前の命令を定義することができない．すでに使われている命令を変更するときには\renewcommand を使って定義する．

11.3 オプション引数を持つ命令

\sqrt という命令は引数を一つ取る場合と二つ取る場合がある．引数が一つの場合，例えば$\sqrt{x}$と書くことで $\sqrt{x}$ が出力される．引数が二つの場合は\sqrt[y]{x}と書くことで $\sqrt[y]{x}$ が出力される．このときの y をオプション引数という．オプション引数は{}ではなく [] でくくって表し，また必ず第一引数の位置に来る．

オプション引数を持つ命令を自分で定義することも出来る．例えば

```
\newcommand{\Xseq}[1][n]{x_1,\dots,x_#1}
```

という定義をしておくと#1 の値にはデフォルトとして n が与えらる．この命令に対して引数を与えずに$\Xseq$と書くと $x_1, \dots, x_n$ が出力される．一方引数を与えるとデフォルトの引数の代わりに，その引数が#1 の値になる．たとえば$\Xseq[k]$と書くと $x_1, \dots, x_k$ が得られる．

複数の引数を持つ命令では第一引数のみオプションにすることが出来る．たとえば

```
\newcommand{\Seq}[2][n]{#2_1,\dots,#2_#1}
```

と定義しておくと，第一引数がオプションで，デフォルトの値は n である．このとき$\Seq{y}$によって $y_1, \dots, y_n$ が得られ，$\Seq[k]{y}$によって $y_1, \dots, y_k$ が得られる．

問題 11.3. (1) 11.2.1 節で定義した ScoreBoard を変更して，ホームチームの名前をオプション引数にしなさい．

(2) 問題 11.2 (2) で定義した MyHeading を変更して，肩書をオプション引数にしなさい．

11.4 環境の定義

環境は基本的に引数を取る命令と変わらないが，引数が複数の行にわたるような長い文章になる場合，環境として定義するのが普通である．

環境は以下のような形式で定義される．

```
\newenvironment{xxx}{yyy}{zzz}
```

xxx は環境の名前である．yyy はその環境の冒頭に置かれ zzz はその環境の末尾に置かれるこ

とになる．たとえば

```
\newenvironment{Proof}{\textbf{証明}．}{\hfill (QED)}
```

と定義したとする[*10]．このとき

```
\begin{Proof}
すべての人間は死ぬ．

ソクラテスは人間だ．

従ってソクラテスは死ぬ．
\end{Proof}
```

と書くと次の出力が得られる．

証明． すべての人間は死ぬ．
ソクラテスは人間だ．
従ってソクラテスは死ぬ． (QED)

環境についても同様にオプション引数を持たせることが出来る．

11.5 自前のパッケージの作成

新しい文書を作るたびに自分で定義した命令をプリアンブルに書きこむのは面倒である．そこで自分で定義した命令を集めたテキストファイルを作り，拡張子を「.sty」としておく．この種類のファイルをスタイルファイルと呼ぶ．仮にこのファイルに「mydefinition.sty」と名前を付けたとする．

mydefinition.sty を，新しい TEX ファイルが置かれているフォルダにコピーしておき，TEX ファイルのプリアンブルに

```
\usepackage{mydefinition}
```

と書くと，mydefinition.sty の中で定義された命令をこの文書中で使うことが出来る．

実際には，スタイルファイルを TEX 関連のファイルが集まっているフォルダの適当な場所に置いておけば，どこからでも参照することができる．しかし後々定義を変更する可能性がある場合はそうしておかない方が無難である．

[*10] `\hfill` はその後ろの文字列を行の右端に寄せる命令である．

11.6 命令を定義することの利点

本節の冒頭で，TEX の最大の利点の一つは命令を定義できることだ，と述べた．命令の定義することの目的は長い命令を短縮するだけものであり，本質的にそれを使わなければ書けない文書というものはない．命令の定義は面倒だし，慣れないうちは思う通りに書けないかもしれない．たくさんの定義をしていくうちに，何をどう定義したのか忘れてしまうこともある．何よりも，同じ表現を何回も使うのであれば，「単語登録とかコピペとか Word や Excel のマクロとか使えばそれで済むじゃん」という意見もあるだろう．しかし TEX での命令の定義には，より大きな利点がある．

プログラミングの世界の格率に，“Don’t Repeat Yourself” というものがある．これは同じパターンの作業を何度も繰り返すな，という教訓である．大きなプログラムの中で，同じパターンの作業が何度も繰り返されるときは，その作業を独立したプログラムとして別に作っておき，それを大きなプログラムの中で呼び出すという形にすることが，以下のような点で有益である．

- 作業効率が良い．
- 別なプログラムの作成にも応用できる．
- プログラムの修正が容易．
- プログラム全体の構造が理解しやすい．

これらのことは TEX での命令の定義にも当てはまる．よく使う命令を定義し，それをスタイルファイルとしてまとめておけば，どの文書からでもそれを呼び出して使うことができる．長い命令を短く定義しておけば文書のソースコードが短く，見やすくなる．さらにコピペしたものは，修正しようと思ったときに，そのすべてに修正を加えなければならないが，定義された命令ならば定義の部分だけを修正すれば，その命令が適用されているすべてのケースが同時に修正されるのである．

たとえば 33 頁の`\ScoreBoard` という命令を考えよう．この命令を何回も使って文書を作ったとする．それから考え直して，表を

	前半	後半	合計
浦和	1	2	3
鹿島	0	1	1

という形式で書きなおしたいと考えたとする．もしもコピペで表を作っていたとすれば，作った表の数だけ修正の作業をしなければならない．つまり 100 個の表を作っていたとすれば 100 回の書き直しが必要なのである．しかしもしもこれを上のような命令として定義していたのであれば，定義を

```
\newcommand{\ScoreBoard}[6]{
\newcount\homesum
\newcount\awaysum
\advance\homesum by #3
\advance\homesum by #5
\advance\awaysum by #4
\advance\awaysum by #6
\begin{tabular}{|l|c|c|c|}
\hline
 & 前半 & 後半 & 合計 \\ \hline
\textbf{#1} & #3 & #5 & \the\homesum \\ \hline
\textbf{#2} & #4 & #6 & \the\awaysum \\ \hline
\end{tabular}
}
```

と書きなおせば良いだけのことである．

このようによく繰り返されるパターンを独立した命令として別に定義しておく，言い換えると作業をモジュール化すること，には大きな利点がある．これはプログラミングや，あるいはより一般的に，工学全般においても重要な考え方である，らしい．

12 文献と索引

12.1 文献の参照

TEX には自動的に参照した文献の一覧を作成する機能がある．そのためにはまず文献のデータベースを作らなければならない．これは「.bib」という拡張子を持つテキスト文書として作成する．まずテキストエディタで「bibtest.bib」という文書を作成し，次のように書き込んでみよう．

```
@book{TakahashiM_Keisan,
  title = "『計算論 --- 計算可能性とラムダ計算』",
  author = "高橋正子",
  publisher = "近代科学社",
  series = "コンピュータサイエンス大学講座",
  address = "東京",
  volume = "24",
  year = "2003",
  yomi = "Takahashimasako"
}
```

このファイルを現在作成している TEX 文書と同じフォルダに保存して置く．

次に現在作成している TEX 文書の中に次のように書いてみよう．

```
高橋\cite{TakahashiM_Keisan}，定理 3.2.17 の証明を参照せよ．
\bibliographystyle{jplain}
\bibliography{bibtest}
```

すると次のように出力されるはずである[*11]．

高橋 [1]，定理 3.2.17 の証明を参照せよ．

参考文献

[1] 高橋正子. 『計算論 — 計算可能性とラムダ計算』, コンピュータサイエンス大学講座, 第 24 巻. 近代科学社, 東京, 2003.

`\cite{xxx}`という命令は，`xxx` という名前の文献を参照することを表している．`\bibliography{yyy}`という命令は `yyy` という名前の bib ファイルから，本文で参照されている文献の書誌情報を探し出して，一定の書式で出力することを命じている．ここの例では bibtest.bib というファイルから，TkahashiM_Keisan という名前の付けられている文献の書誌データをとりだし，出力している．本文で参照されている文献には，自動的に番号がつけられ，`\cite` が書かれている個所には，その番号が出力される．

文献には単行本，シリーズの一巻，論文集に収められた論文，定期刊行誌に収められた論文など，様々な種類があり，その種類に応じて必要な書誌情報，出力のスタイルが変わってくるのであるが，ここでは詳しくは述べず，例をあげるに留めよう．

[*11] 高橋 [?] のように出力されるときは，何度かコンパイルを繰り返す．

```
@incollection{Godel1,
 title = "What is {Cantor's} continuum problem",
 author = "G{\"{o}}del, K.",
 booktitle = "Philosophy of Mathematics: Selected Readings",
 publisher = "Cambridge University Press",
 address = "Cambridge",
 year = "1983",
 editor = "Benacerraf, P. and Putnam, H.",
 edition = "Second",
 pages = "470--485"
}
@article{Ichise1,
 title = "「帰納学習における帰納論理プログラミングと遺伝的プログラミングの統合」",
 author = "龍太郎，市瀬 and 正行，沼尾",
 journal = "『人工知能学会誌』",
 year = "1999",
 volume = "14",
 pages = "307--314",
 yomi = "Iichise"
}
```

これらを参照した場合，文献リストは次のようになる．

参考文献

[1] K. Gödel. What is Cantor's continuum problem. In P. Benacerraf and H. Putnam, editors, *Philosophy of Mathematics: Selected Readings*, pp. 470–485. Cambridge University Press, Cambridge, second edition, 1983.

[2] 市瀬龍太郎, 沼尾正行. 「帰納学習における帰納論理プログラミングと遺伝的プログラミングの統合」. 人工知能学会誌, Vol. 14, pp. 307–314, 1999.

[3] 高橋正子. 『計算論 — 計算可能性とラムダ計算』, コンピュータサイエンス大学講座, 第 24 巻. 近代科学社, 東京, 2003.

文献参照のスタイルには様々あるがここでは詳細には立ち入らない．詳しくは奥村晴彦の前掲書などを参照されたい．また標準で備わっていないスタイルも Web などで手に入るものがたくさん

ある．

手打ちで文献表を作るのはかなりの労力なので，普段から本や論文を読んだ時はデータベースに登録しておくと便利である．