LaTeX マクロ入門

荒田 実樹

2018 年 4 月 26 日

目次

1. マクロとは
2. 自分でマクロを定義しよう
3. 環境
4. \def vs \newcommand

マクロとは

- 命令列を別の命令列に置き換える
- よく使う命令列に短い名前を与えることができる
- 例えば、\TeX というマクロは
  `T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX\@`
  として定義されている

コマンドとマクロの関係

- コマンド：TEX 処理系や文書への命令
- マクロ：コマンドのうち、別の命令列に置き換わるもの
- TEX primitive：コマンドのうち、TEX 処理系が直接処理するもの
- LATEX や各種パッケージが提供するコマンドは全てマクロとして実装されている
- 使う側としては、特定のコマンドがマクロか TEX primitive かは意識しなくて良い

TEX primitive の例

`\def`, `\par`, `\displaystyle`, `\jobname`, `\left`, `\right`, など

マクロとして定義されたコマンドの例

`\TeX`, `\frac`, `\begin`, `\item`, 他多数

コマンド名について

- 基本：バックスラッシュ\の後にアルファベットを並べる
  - このパターンでは数字や記号は使えない
  - TEX 処理系が日本語対応の場合は、アルファベットの他に漢字や仮名も使える
  - 例：\TeX, \section, \西暦
  - このパターンの場合、直後の空白文字は無視される
    例：“\TeX primitive” は “TEXprimitive” となる（cf. \TeX\ primitive）
- バックスラッシュの後に記号 1 文字
  - 例：\!, \,
  - 直後の空白文字は無視されない
- 特殊文字の一部：（標準では）半角チルダ~
  - 利用者的には「コマンド名」というよりも「特殊文字」というくくりの方が自然かもしれない

マクロの使い方 1

- コマンド名の後に引数を書く
- 必須の引数は波カッコ{ }で囲む
  - 波カッコ{ }は入れ子にできる
  - 囲むために使った波カッコ自身は引数には含まれない
- 必須の引数が 1 文字またはコマンド名 1 個の場合は{ }を省略できる
  - `\frac12` は`\frac{1}{2}`と等価
  - `\newcommand\Real{\mathbf{R}}`
    は
    `\newcommand{\Real}{\mathbf{R}}`
    と等価
  - 「省略できる」からといって省略するかどうかは、書く人の美意識次第だが…

マクロの使い方 2

- 省略可能（オプショナル）な引数を与える場合は角カッコ [ ] で囲む
  - 例：`\sqrt[5]{\pi}`
  - 角カッコ [ ] は直接入れ子にはできないが、一旦波カッコで囲めば入れ子にできる
- 一部のコマンドは、コマンド名の直後にスターを置くことによって挙動が変わる
  - 例：`\section*`, `\newcommand*`, `\verb*`
  - 「スターもコマンド名の一部」のように見えるが、実際は違うので要注意

マクロを定義するコマンド：\newcommand

- `\newcommand{コマンド名}{内容}`

例

- `\newcommand{\Real}{\mathbb{R}}`
- `\Real` コマンドは`\mathbb{R}`に置き換わる
- マクロのメリット
  - TEX ソース中に「見た目」だけではない「意味」を記述できる
  - 見た目を変更したくなった時に、マクロの定義だけを変えれば良い
    例：$\mathbb{R}$ ではなく $\mathbf{R}$ にしたくなったら、`\Real` の定義を`\mathbf{R}`に変えるだけで済む
- 注意点：複数人で作業する場合に各自が好き勝手にマクロを定義すると収拾がつかなくなる

引数を取るコマンドを定義する

- \newcommand{コマンド名}[引数の個数]{内容}
- コマンド名と内容の間に、オプション引数として引数の個数を渡す
- 「内容」の中に書いた#1〜#9 が実際の引数に置き換えられる（引数の個数は最大で 9 個）

例

- \newcommand{\transpose}[1]{{}^t\,#1}
- \transpose{A}が{}^t\,A に置き換えられる
- マクロのメリット：転置の記法を変えたくなった場合は、\transpose の定義を変えれば良い

引数を省略可能にする

- `\newcommand{コマンド名}[引数の個数][デフォルト値]{内容}`
- デフォルト値を指定することで、最初の引数（`#1`）が省略可能になる
- 2 番目以降の引数は省略可能にできない（`\newcommand` の弱点）

例

- `\newcommand{\norm}[2][2]{\left\lVert #2\right\rVert_{#1}}`
- `\norm{x}`が $\|x\|_2$ に、`\norm[\infty]{x}`が $\|x\|_\infty$ になる

（上級）長い引数と短い引数

- 長い引数：引数の中に空行を含められる
- 短い引数：引数の中に空行を含められない
  - 目的：波カッコ } の閉じ忘れのミスを発見しやすくする
  - TEX 的には空行は\par コマンドと等価なので、短い引数の中に\par は書けない
- \newcommand の直後にスター*を置くと、定義するマクロの引数が短い引数になる
- **\newcommand***{コマンド名}[引数の個数] [デフォルト値]{内容}

例（閉じカッコ忘れ）

```
\section{この TA 小話がすごい！ 2018

2018 年の大賞はこれだ！！！
```

というソースをコンパイルすると Paragraph ended before ... was complete というエラーが出る

マクロの上書き

- \renewcommand を使うと、定義済みのコマンドを上書きできる
- 使い方は\newcommand と同様：
  \renewcommand{コマンド名}[引数の個数] [デフォルト値]{内容}

例：\Re コマンドを上書きする

- 標準では\Re は $\Re$
- **\renewcommand**{\Re}{\operatorname{Re}}とすると
- Re になる

マクロのスコープについて

- 定義したマクロはどの範囲で有効か？
- スコープが区切られる条件：
  - 単体の波カッコ `{ }`（マクロ引数としての `{ }` は該当しない）
  - 環境（`\begin{}` .. `\end{}`）
- `\renewcommand` で上書きしても、スコープの外に出ると上書きした定義は忘れられる

マクロのスコープについて

例

```
\begin{enumerate}
  % \labelenumi コマンドの定義を上書きする
  \renewcommand{\labelenumi}{(\theenumi)}
  \item hoge
  \item piyo
\end{enumerate}

% \labelenumi の定義は元に戻っている
\begin{enumerate}
  \item foo
  \item bar
\end{enumerate}
```

環境とは

- \begin .. \end で囲むやつ
- 文書の構造を表現する手段の一つ
- 環境は、LaTeX 特有の概念である（plain TeX には環境という概念はない）
- 環境の名前はコマンド名と違い、アルファベットの他に、数字やアスタリスクを含めることができる
- マクロと同様に、環境は引数を受け取れる。使う際は\begin{環境名}の直後に引数を与える

例：array 環境

```
\begin{array}{cccc}
  f\colon & A & \to & B \\
          & x & \mapsto & y
\end{array}
```

\begin{array}直後の{cccc}が環境への引数

環境を定義する

- 環境は\newenvironment コマンドで定義できる
- 基本形：**\newenvironment**{名前}{開始}{終了}
- 引数を取る場合：**\newenvironment**{名前}[引数の個数] [デフォルト値]{開始}{終了}
  - 引数の指定については\newcommand と同様
  - 「開始」の中にある#1〜#9 が引数で置き換えられる
  - 「終了」の部分では引数は参照できない

マクロと環境の関係

- `\begin{hoge}`〜`\end{hoge}`
  は大雑把には
  `{\hoge`〜`\endhoge}`
  と等価

重要

環境と同じ名前のコマンドを共存させることはできない

\def vs \newcommand

- LaTeX においてマクロを定義する方法として\def と\newcommand の 2 種類が使われているが、どう違うのか？
- **\def**\Real{\mathbb{R}}
  か
  \newcommand{\Real}{\mathbb{R}}
  か？

使い方の違い

- \def(コマンド名) (引数の指定){内容}
- \newcommand{コマンド名}[引数の個数][デフォルト値]{内容}
- \def はマクロではないので、コマンド名を{ }で括ることはできないし、内容を囲む{ }を省略できない
- \def では省略可能な引数は簡単には定義できず、TEX プログラミングが必要となる

挙動の違い

- 既存の同名のコマンドまたは環境があっても、\def は問答無用で上書きする
  - →意図せずに既存のコマンドや環境を壊す可能性がある。危険！
  - （常識的な範囲ではそのような「事故」はそうそう起こらないと思うが…）
- \newcommand は既存のコマンド・環境があるとエラーになる
  - →意図せずに既存のマクロや環境を壊すことがない。安全！
  - 上書きしたい場合は\renewcommand を使う
- \newcommand では一部の名前を定義できない（end から始まる名前及び relax）
  - 安全のため

\newcommand では物足りない！

- 「ぼくのかんがえたさいきょうの数学記法を LaTeX で実現するには\newcommand で定義できる形式では不足だ！」
- xparse パッケージで提供される\NewDocumentCommand 系のコマンドを使うと、「スターの有無で挙動を変える」「複数の省略可能引数を持つ」など、より柔軟なマクロを定義できる
- 詳しくはググるか texdoc xparse

過去のスライド

私（荒田）が過去の計算数学で話した TA 小話の資料

https://github.com/minoki/ks-slide

「TEX の字句解析」「浮動小数点数」など