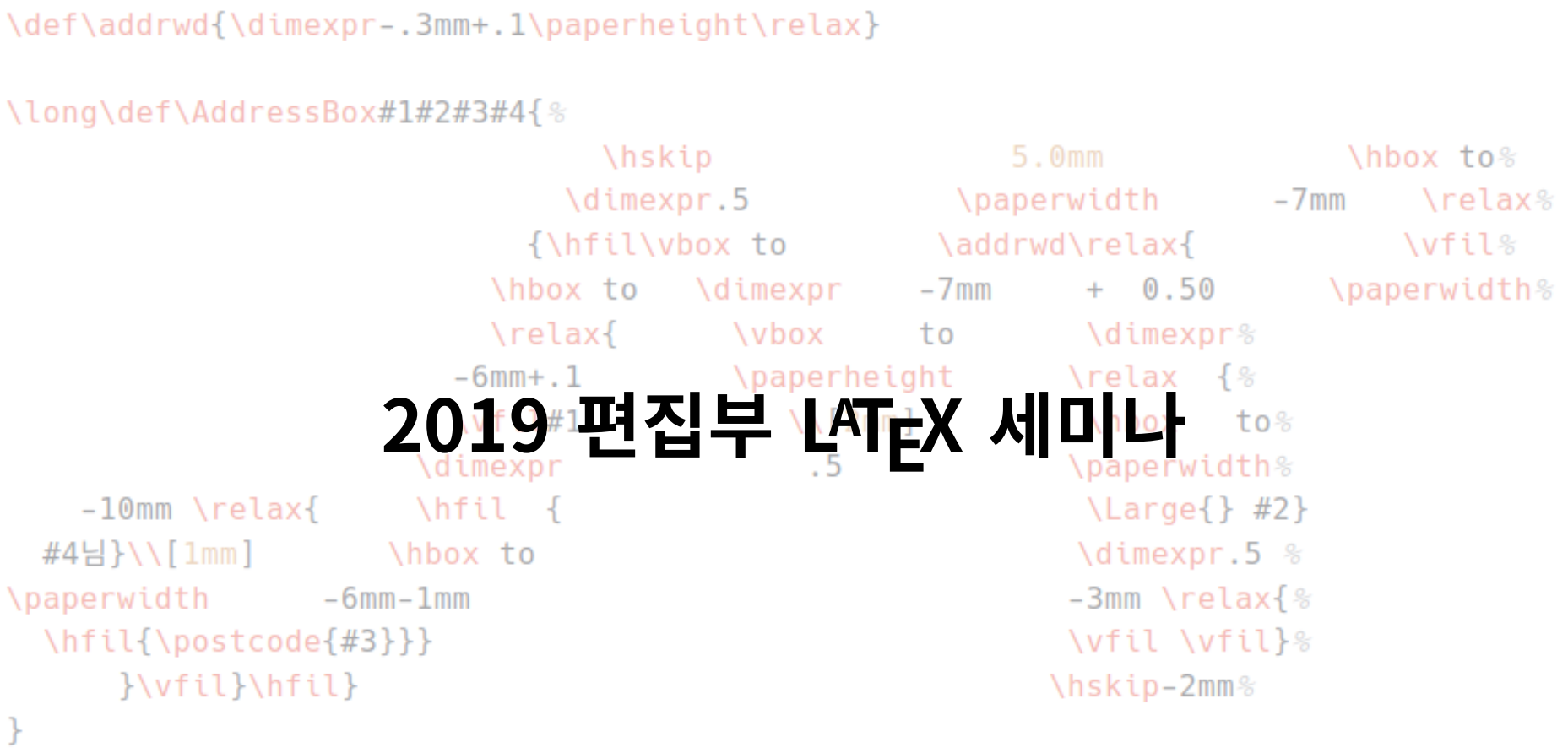
2019 편집부 LaTeX 세미나



Introduction

LaTeX 설치하기

- TeX은 여러 가지 배포판이 있습니다.
- TeX Live 배포판 설치하기 (가장 쉬운 방법)
  - `https://www.tug.org/texlive/`에 방문하여 설치
  - `texlive-full`을 받으면 정신건강에 매우 편하지만, 약 **4.9 GB**를 차지합니다.
  - `texlive-basic`을 받으면 초기 용량을 **64 MB**까지 줄일 수 있지만, 필요한 패키지를 일일이 설치해야 합니다.
  - 정식 release는 2018까지 있고, 현재 2019 preview가 진행되고 있습니다. `https://tug.org/texlive/pretest.html`
- MikTeX (Windows) (`https://miktex.org/`)
- TnXTeX (`http://wiki.ktug.org/wiki/wiki.php/TnXTeX`)
- TinyTeX (`https://yihui.name/tinytex/`)

MikTeX은 TeX Live 기반 배포판과 설정 방법이 달라 추천드리지 않습니다.

TeX Live 패키지 설치하기 (스킵 가능)

- texlive-full이 아닌 더 적은 패키지를 포함하고 있는 버전을 받으셨다면, 추가로 설치해야 하는 패키지들이 있을 수 있습니다.
- 문서를 작성하다가 `File `~~~~.sty' not found.`라는 에러 메시지가 발생하면, 다음 명령어를 터미널 (혹은 cmd)에 입력해 주세요.
  - `tlmgr search --global --file "/패키지_이름.sty"`를 입력하면
  - '패키지_묶음_이름: texmf-dist/.../패키지_이름.sty'라는 문구를 볼 수 있고,
  - `tlmgr install 패키지_묶음_이름`을 치면 설치됩니다.
- 다운로드 받은 sty 파일을 설치하고 싶다면, https://github.com/msquare-kaist/mathletter-package/blob/master/documents/manual.pdf를 참조해 주세요.

TEX이란?

- TEX: Donald Knuth가 개발한 문서 조판 도구
- LATEX: Leslie B. Lamport가 개발한 TEX의 확장 (매크로 모음)
  - LATEX 2_ε: 가장 많이 쓰이는 LATEX 버전
  - LATEX 3: 현재 개발 중인 LATEX 버전
- 이외에도 ConTEXt이라는 확장이 있습니다.
- Overleaf: 가장 유명한 온라인 LATEX 동시 편집 서비스 (`https://overleaf.com`)

Token

- TEX 문서는 토큰들로 이루어져 있으며, TEX 엔진이 각 토큰을 **전개** (expand) 하면서 실행됩니다.
- **매크로** (macro) 또는 **정의** (definition) 란 기본적으로 정의되어 있는 토큰들 (primitives) 을 이용하여 정의된 토큰들을 말합니다.
- 매크로는 보통 `\macro` 와 같이 백슬래시로 시작하고 알파벳이 뒤에 따라 옵니다.
- 한 글자짜리 매크로는 알파벳이 아니어도 올 수 있습니다. `\ `, `\#`, ...
- {...}는 여러 토큰을 하나처럼 묶어주는 토큰.

문서 구조

- `\documentclass[a4paper,11pt]{article}` 문서의 논리적 구조를 지정합니다.
  - book, report, memoir 등의 class가 있습니다.
- `\usepackage{amsmath}` 미리 정의된 매크로를 **패키지**에서 가져옵니다.
  - 사실은 `\usepackage` 는 `\RequirePackage` 로 정의됩니다.
- `\title{...}` , `\author{...}` , `\date{...}`
- `\begin{document}` 문서를 시작합니다.
- `\maketitle` 제목, 저자, 날짜를 출력합니다.
- `% 주석입니다` 퍼센트 기호 뒤는 무시됩니다.
- `\end{document}` 문서를 끝냅니다.

Sample Document

따라만 하세요 따라만 하세요

```
\documentclass[a4paper,10pt]{article}
\usepackage{amsmath}
\usepackage{kotex}
\title{제목}

\begin{document}
\maketitle

안녕하세요? \LaTeX을 배우면 좋은 일이 있을 거예요. % 아닐 수도..

\makeatletter
당신이 입력했던 제목입니다: \@title.\footnote{신기한가요??}
\makeatother
\end{document}
```

글씨체

인자를 받는 매크로와 스위치

- 글씨체를 바꾸는 매크로는 크게 두 가지로 나뉩니다.
- 인자를 받는 매크로: `{...}` 안에 있는 글씨만 바꿉니다. `\text`나 `\math` 등으로 시작합니다.
  - `\textit{italic here only} not here`
  - *italic here only* not here
- 스위치: 어떤 scope 안에서, 그 매크로 이후의 글씨에 **모두 적용**됩니다. 사용에 주의해 주세요. `\...shape`나 `\...series`, `\...family`로 끝납니다.
  - `\itshape{italic here and} also here`
  - *italic here and also here*

글씨체

일반 텍스트의 글씨체

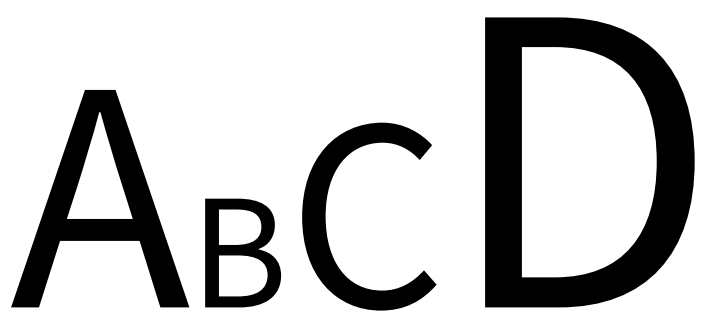
- `{\normalfont 기본 폰트}` 기본 폰트
- `\textbf{굵게 bold}`, `{\bfseries 굵게 bold}` **굵게 bold**
- `\textit{이탤릭 italic}`, `{\itshape 이탤릭 italic}` *이탤릭 italic*
  - `\textit` 는 그 다음에 똑바로 선 글자가 왔을 때 간격 보정을 해줍니다.
  - `\textit{ital}ic`, `{\itshape ital}ic`
  - *italic*
  - *italic*
- `\textrm{명조 roman}`, `{\rmfamily 명조 roman}` 명조 roman
- `\textsf{돋움}`, `{\sffamily sans-serif}` 돋움 sans-serif
- `\textsl{기울임}`, `{\slshape slanted}` *기울임 slanted*
- `\textsc{SmallCaps}`, `{\scshape SmallCaps}` SmallCaps
- `\texttt{typewriter}`, `{\ttfamily typewriter}` typewriter

글씨체

수식 모드의 글씨체

- `\mathbf{Bold}` $\mathbf{Bold}$
- `\boldsymbol{\alpha}` $\boldsymbol{\alpha}$ (알파벳이나 숫자가 아닌 기호를 굵게)
- `\mathit{italic}` italic
- `\mathrm{Roman}` Roman
- `\mathsf{Sans-serif}` $\mathsf{Sans-serif}$
- `\mathbb{BLACK}` $\mathbb{BLACK}$ (대문자만)
- `\mathcal{CALI}` $\mathcal{CALI}$ (대문자만)
- `\mathscr{SCRIPT}` $\mathscr{SCRIPT}$ (대문자만)
- `\mathfrak{Fraktur}` $\mathfrak{Fraktur}$

글씨 크기

- `\tiny \scriptsize \footnotesize \small \normalsize \large \Large \LARGE \huge \Huge`
- AAAAAAAAA
- `\fontsize{글씨 크기}{line height}\selectfont`
  - 보통 line height는 글씨 크기의 1.2 배
- 

수식 입력하기

- `$ ... $` 다른 글씨들과 같이 출력되는 수식입니다. 예) $\alpha^2\beta$
  - `\( ... \)`을 쓸 수도 있지만, 이 매크로는 **fragile**한 매크로라 다른 매크로의 인자로 들어가면 오류를 내므로 사용하지 않는 편이 낫습니다.
- `\[ ... \]` 한 줄 전체를 차지하는 수식입니다.
  - `$$ ... $$`을 쓸 수도 있지만, 이 매크로는 여백 조절이나 수식 넘버링, 또 에러 메시지 출력 등에서 위의 것보다 안 좋습니다.
- 기본적인 수식 매크로:

```
$ \alpha, \beta, \frac{\sqrt{2}}{3}, \cdots $
```

  - $\alpha, \beta, \frac{\sqrt{2}}{3}, \cdots$
- 더 많은 매크로와 기호는
  - https://www.codecogs.com/latex/eqneditor.php나
  - http://detexify.kirelabs.org/classify.html에서!

수식 매크로는 많이 외우셨으면 좋겠습니다

```
\documentclass[a4paper,10pt]{article}
\usepackage{amsmath,amsfonts,amssymb,mathrsfs,mathtools,kotex}
\title{\scshape Practice}
\begin{document}
\maketitle
\[ \text{근의 공식}:\qquad x = \frac{-b\pm\sqrt{b^2 - 4ac}}{2a}. \]
$\alpha, \beta$는 있지만 \texttt{\char`\\Alpha, \char`\\Beta}는 없다!

Radical of an ideal $\sqrt{\mathfrak a}$

크기가 조절되는 괄호를 열 때에는 $\left( \frac a b \right)$처럼 씁니다.
강의할 때 어느 정도 설명해주세요 \char`\^\char`\^7
\end{document}
```

^^7

공백, 행간 간격

- 문단을 띄울 때에는 엔터를 두 번 치거나 `\par` 를 입력합니다.
- 강제로 한 줄을 띄울 때에는 `\\` 를 입력합니다.

공백, 행간 간격

- 문단을 띄울 때에는 엔터를 두 번 치거나 `\par` 를 입력합니다.
- 강제로 한 줄을 띄울 때에는 `\\` 를 입력합니다.
- `\! \, \ \quad \qquad ~` 는 텍스트 모드에서의 공백을 나타냅니다. 크기는 다음과 같습니다.
  - `a a\!a\,a\ a\quad a\qquad a~a` → a aa a a a a a
  - `\!` 는 간격을 줄입니다.
  - `~` 는 끊어지지 않는 공백을 나타내는 'active character'입니다.
- `\! \, \ \> \: \; \quad \qquad ~` 는 수식 모드에서의 공백을 나타냅니다. (직접 테스트 해보세요)

공백, 행간 간격

- 문단을 띄울 때에는 엔터를 두 번 치거나 `\par` 를 입력합니다.
- 강제로 한 줄을 띄울 때에는 `\\` 를 입력합니다.
- `\! \, \ \quad \qquad ~` 는 텍스트 모드에서의 공백을 나타냅니다.
  크기는 다음과 같습니다.
  - `a a\!a\,a\ a\quad a\qquad a~a` → a aa a a a a a
  - `\!` 는 간격을 줄입니다.
  - `~` 는 끊어지지 않는 공백을 나타내는 'active character'입니다.
- `\! \, \ \> \: \; \quad \qquad ~` 는 수식 모드에서의 공백을 나타냅니다. (직접 테스트 해보세요)
- `\hspace{길이}` , `\hspace*{길이}` 는 가로로 ⟨길이⟩만큼의 간격을 만듭니다.
  원래는 한 줄을 시작할 때엔 무시되는데, *이 붙으면 살아납니다.
- `\vspace{길이}` , `\vspace*{길이}` 는 세로 버전

공백, 행간 간격

- 문단을 띄울 때에는 엔터를 두 번 치거나 `\par` 를 입력합니다.
- 강제로 한 줄을 띄울 때에는 `\\` 를 입력합니다.
- `\! \, \ \quad \qquad ~` 는 텍스트 모드에서의 공백을 나타냅니다.
  크기는 다음과 같습니다.
  - `a a\!a\,a\ a\quad a\qquad a~a` → a aa a a a a a
  - `\!` 는 간격을 줄입니다.
  - `~` 는 끊어지지 않는 공백을 나타내는 'active character'입니다.
- `\! \, \ \> \: \; \quad \qquad ~` 는 수식 모드에서의 공백을 나타냅니다. (직접 테스트 해보세요)
- `\hspace{길이}` , `\hspace*{길이}` 는 가로로 ⟨길이⟩만큼의 간격을 만듭니다.
  원래는 한 줄을 시작할 때엔 무시되는데, *이 붙으면 살아납니다.
- `\vspace{길이}` , `\vspace*{길이}` 는 세로 버전
- `\noindent` 는 문단 시작 시 들여쓰기를 없앱니다.

공백 관련

- ★ TEX은 띄어쓰기에 민감합니다. 예를 들어 괄호 앞뒤로 띄어쓰기를 하는지가 상관없는 C나 Python 등 다른 언어와 달리, TEX 문서를 쓸 때엔 띄어쓰기를 주의해야 합니다.

공백 관련

- TEX은 띄어쓰기에 민감합니다. 예를 들어 괄호 앞뒤로 띄어쓰기를 하는지가 상관없는 C나 Python 등 다른 언어와 달리, TEX 문서를 쓸 때엔 띄어쓰기를 주의해야 합니다.
- 텍스트 모드에서, 띄어쓰기는 잘 없어지지 않습니다. 띄어쓰기가 없어지는 경우는:
  - 두 개 이상 연속 띄어쓰기가 있을 때 한 개만 남습니다. 즉 띄어쓰기 여러 개를 쓰려면 `\ \ \` 와 같이 써야 합니다.
  - 한 줄의 맨 앞 (leading) 에 있는 띄어쓰기는 모두 무시됩니다.

공백 관련

- TEX은 띄어쓰기에 민감합니다. 예를 들어 괄호 앞뒤로 띄어쓰기를 하는지가 상관없는 C나 Python 등 다른 언어와 달리, TEX 문서를 쓸 때엔 띄어쓰기를 주의해야 합니다.
- 텍스트 모드에서, 띄어쓰기는 잘 없어지지 않습니다. 띄어쓰기가 없어지는 경우는:
  - 두 개 이상 연속 띄어쓰기가 있을 때 한 개만 남습니다. 즉 띄어쓰기 여러 개를 쓰려면 `\ \ \` 와 같이 써야 합니다.
  - 한 줄의 맨 앞 (leading) 에 있는 띄어쓰기는 모두 무시됩니다.
- 텍스트 모드에서, 줄바꿈의 위아래 줄이 비어있지 않으면 그 줄바꿈은 띄어쓰기로 처리됩니다.
  - 이 띄어쓰기를 없애려면, 코드 가독성을 위해 줄바꿈을 했지만 띄어쓰기는 원하지 않는다면, 그 줄 끝에 주석 기호 %를 붙이면 그 줄바꿈이 주석 처리되어 사라집니다. 참고로, 다음 줄의 앞에 있는 띄어쓰기까지 모두 사라집니다.

공백 관련

- 수식 모드에서는 띄어쓰기가 잘 영향을 주지 않습니다. 따라서 수식 안에서 텍스트 모드를 실행하려면, `\text{...}` 등으로 감싸 주어야 합니다.
  - `$not a good example$` $notagoodexample$
  - `$\textrm{\textit{good example}}$` $good\ example$

달러나 백슬래시는 어떻게 입력할까요

미국인이 만들었는데 달러를 입력 못할 리가 없겠죠!!

- `\& \% \$ \# \_ \{ \} \char`\~\␣\char`\^\␣\char`\\`
- & % $ # _ { } ~ ^ \

달러나 백슬래시는 어떻게 입력할까요

미국인이 만들었는데 달러를 입력 못할 리가 없겠죠!!

- `\& \% \$ \# \_ \{ \} \char`\~\␣\char`\^\␣\char`\\`
- & % $ # _ { } ~ ^ \
- 살짝씩 다르지만 다른 입력 방법도 많습니다:
- `\textasciitilde\␣\textasciicircum\␣\textbackslash`
- ~ ^ \

달러나 백슬래시는 어떻게 입력할까요

미국인이 만들었는데 달러를 입력 못할 리가 없겠죠!!

- `\& \% \$ \# \_ \{ \} \char`\~\␣\char`\^\␣\char`\\`
- & % $ # _ { } ~ ^ \
- 살짝씩 다르지만 다른 입력 방법도 많습니다:
- `\textasciitilde\␣\textasciicircum\␣\textbackslash`
- ~ ^ \
- `\^{} \~{} \string^ \string~ \string\`
- ^ ~ ^ ~ \

Environments

환경

Environments

- `\begin{ENVNAME} ... \end{ENVNAME}` 꼴의 매크로를 모두 환경이라고 부릅니다.
  - 사실 이것은 `\ENVNAME ... \endENVNAME` 로 정의됩니다.
- 여러 중요한 환경들:
  - `itemize`와 `enumerate` (with `enumitem`)
  - `matrix`
  - `amsthm`
  - `align`
  - `figure`와 `center`
  - `tabular`

itemize와 enumerate

예시만 넣어 놓으면 택님께서 설명해 주실 거예요

enumitem 패키지를 불러온 뒤,

```
\begin{itemize}
  \item 첫 번째
  \item[2] 두 번째
  \item[$\gamma$] wow
\end{itemize}

\begin{enumerate}[label={\roman*\arabic*}]
  \item 자동으로
  \item 올라가는
  \item 번호
\end{enumerate}
```

matrix

예시만 넣어 놓으면 택님께서 설명해 주실 거예요

수식 모드 안에서,

```
\begin{itemize}
\[
  \begin{pmatrix}
    a_{11} & a_{12} \\
    a_{21} & a_{22}
  \end{pmatrix} \cdot
  \begin{bmatrix}
    b_{11} & b_{12} \\
    b_{21} & b_{22}
  \end{bmatrix} =
  \begin{vmatrix}
    c_{11} & c_{12} \\
    c_{21} & c_{22}
  \end{vmatrix} \cdot \mathbf I
\]
```

amsthm

예시만 넣어 놓으면 택님께서 설명해 주실 거예요

amsthm 패키지를 불러온 뒤,

```
\newtheorem{thm}{Theorem} % basic usagae
\newtheorem{lem}[thm]{Lemma} % shares the numbering w/ thm

\theoremstyle{definition}
\newtheorem{defn}{Definition}[section] % Def section.def_number

\theoremstyle{plain} % recover the theoremstyle
\newtheorem*{cor}{Corollary} % no numbering (*)

\begin{lem}[Lemma Name] This lemma is very awesome. \end{lem}
\begin{proof} Leave as an exercise. \end{proof}
```

figure와 center

예시만 넣어 놓으면 택님께서 설명해 주실 거예요

float, graphicx, (adjustbox) 패키지를 불러온 뒤,

```
% H means that it tries to put on the exact position
\begin{figure}[H]
  \begin{center}
    \includegraphics[width=0.5\textwidth]{msquare}
    \caption{Gorgeous}
  \end{center}
\end{figure}
```

tabular

예시만 넣어 놓으면 택님께서 설명해 주실 거예요

https://www.tablesgenerator.com/

Macros / Definitions

매크로

귀찮은 일은 싫어하는 사람들의 모임

- 편집부장님: “레이텍으로 구구단을 만들어 주세요!”
- 안타까운 사람들: `$2 \times 2 = 4$, $2 \times 3 = 6$, ...`

매크로

귀찮은 일은 싫어하는 사람들의 모임

- 편집부장님: “레이텍으로 구구단을 만들어 주세요!”
- 안타까운 사람들: `$2 \times 2 = 4$, $2 \times 3 = 6$, ...`
- 매크로를 배운 후:

```
\newcounter{left} \newcounter{right}
\forloop{left}{2}{\value{left} < 10}{%
  \noindent\forloop{right}{2}{\value{right} < 10}{%
    $\arabic{left} \times \arabic{right} =
      \the\numexpr\arabic{left} * \arabic{right}\relax$,
  }
  \par
}
```

기본적인 매크로

plain TEX버전

- `\def\mymacro#1#2#3{\textit{#1}#2\textrm{#3}}`
  - `\def` 매크로 정의를 시작하는 토큰
  - `\mymacro#1#2#3` 매크로 이름과 parameter 개수 (<9), 즉 parameter를 `\mymacro{p1}{p2}{p3}` 처럼 받는다는 이야기
  - `{\textit{#1}#2\textrm{#3}}` 매크로 이름이 나온 후 처음으로 {를 만나면 그 매크로를 전개했을 때 나오는 매크로의 '내용'을 정의하게 됩니다.
- 다른 기호를 쓸 수도 있어요!! `\mymacro[#1]#2\middle #3{#1#2#3}` 패턴 매칭처럼 행동합니다. 이 방법은 plain TEX에서 가끔 배열을 다뤄야 할 때 유용해요.
- 한글은 쓰지 마세요...

기본적인 매크로

LaTeX 2ε 버전

- `\newcommand{\mymacro}[3][A]{#1,#2,#3}`
  - `\newcommand` 매크로 정의를 시작하는 토큰
  - `{\mymacro}` 매크로 이름
  - `[3]` parameter 개수 (<9)
  - `[A]` #1의 기본값 (기본값은 가장 처음 인자 한 개까지만 지정 가능하며, 기본값이 있는 optional한 인자는 `{...}`가 아닌 `[...]`로 묶어주어야 합니다.)

기본적인 매크로

LaTeX 3 버전

- `\NewDocumentCommand{\mymacro}{s m o O{3}}{#1,#2,#3,#4}`
  - `\usepackage{xparse, expl3}` LaTeX 3 관련
  - `\NewDocumentCommand` 매크로 정의를 시작하는 토큰
  - `{\mymacro}` 매크로 이름
  - `{s m o O{3}}` parameter 개수와 그 spec
    - `m`: 필수 인자
    - `o`: 기본값 없는 optional한 인자; 기본값 없는 optional 인자에 값이 주어지지 않으면 대신 `-NoValue-`라는 값이 들어갑니다.
    - `O{3}`: 기본값이 3인 optional한 인자
    - `s`: 별표가 있는지 없는지 감지
    - 다른 spec은 패키지 `xparse` 매뉴얼 참조
- 지금까지 나온 매크로들은 텍스트와 수식 모드 모두에서 쓸 수 있습니다.

매크로 예시!!

간단하고 쉬운 것

- α를 간단하게 쓰고 싶을 때:
- `\newcommand{\bsalpha}{\boldsymbol{\alpha}}` $\boldsymbol{\alpha}$
- $\mathbb{Z}$를 간단하게 쓰고 싶을 때:
- `\newcommand{\ZZ}{\mathbb{Z}}` $\mathbb{Z}$

매크로 예시!!

간단하고 쉬운 것

- 벡터 표기하기
  - `$\vecNotation{X}$` 를 하면 $(X_1, X_2, X_3)^\intercal$가 나오게 하고 싶다!
  - `$\vecNotation{X}[1]$` 를 하면 (X_1)가 나오게 하고 싶다!
  - `$\vecNotation{X}[2]$` 를 하면 $(X_1, X_2)^\intercal$가 나오게 하고 싶다!
  - `$\vecNotation{X}[4]$` 를 하면 $(X_1, X_2, \ldots, X_4)^\intercal$가 나오게 하고 싶다!
  - `$\vecNotation{X}[n]$` 를 하면 $(X_1, X_2, \ldots, X_n)^\intercal$가 나오게 하고 싶다!

매크로 예시!!

간단하고 쉬운 것

```
\ExplSyntaxOn % _와 :를 매크로 이름에 쓸 수 있게 만듭니다.
\NewDocumentCommand{\vecNotation}{m O{3}}{
  % \tl_if_eq:nnTF{토큰 리스트1}{토큰 리스트2}{같을 때}{다를 때}
  \tl_if_eq:nnTF{#2}{1}{
    (#1\sb 1) % \ExplSyntaxOn이 _를 문자로 바꾸므로
              % \sb라는 명령어로 subscript를 만듭니다.
  }{
    \tl_if_eq:nnTF{#2}{2}{
      (#1\sb 1, #1\sb 2)^{\mathsf{T}}
    }{
      \tl_if_eq:nnTF{#2}{3}{
        (#1\sb 1, #1\sb 2, #1\sb 3)^{\mathsf{T}}
      }{
        (#1\sb 1, #1\sb 2, \dots, #1\sb{#2})^{\mathsf{T}}
      }
    }
  }
}
\ExplSyntaxOff % _와 :를 다시 각자의 목적으로 되돌립니다.
```

Counter

For LaTeX 2ε

- 정수(integer)형 변수 역할
- `\newcounter{CounterName}` 새 카운터를 만듭니다.
- `\setcounter{CounterName}{value}` 카운터 값 지정
- `\addtocounter{CounterName}{value}` 카운터에 값을 더하기 (음수도 가능)
- `\value{CounterName}` 카운터 값 출력

For loop

`forloop`, `pgffor`

- `\forloop[step]{counter}{initial}{condition}{code}`
  - `\usepackage{forloop}`
  - `\newcounter{ct}`

    ```
    \forloop{ct}{1}{\value{ct} < 10}{\arabic{ct} }
    ```
  - 1 2 3 4 5 6 7 8 9␣
- `\foreach \macro in {1,4,...,10}{...}`
  - `\usepackage{pgffor}`
  - ```
    \newcommand{\repsum}[3]{%
      \foreach \i in {1,...,#1}{
        \ifnum\i>1
          + #2_{\i} #3_{\i}
        \else
          #2_{\i} #3_{\i}
        \fi
      }
    }
    ```
    ```

커버하지 않은 내용

너무 많아요

\label\ref\url\definecolor\textcolor\color\newenvironment
\expandafter\noexpand\IfNoValueTF\IfBooleanTF\makeatletter
\makeatother\@author\@date\begin{tikz}...\end{tikz}\edef
\@gobble\relax\dimexpr\numexpr \the\meaning\global
\xdef\csname...\endcsname\@firstofone\z@\@car\@cdr\@nil
\toks@\loop\@for\g@addto@macro\parskip\parindent\protected
\unexpanded\renewcommand\RedeclareSectionCommand\hbox\pbox
\rule\raisebox\kern\hskip\vskip\glueexpr\allowdisplaybreaks,...

패키지: expl3, xkeyval, pgf, tikz, tikzcd, ulem, tcolorbox, bibtex, natbib, fancyhdr, tcolorbox, environ, listings, minted, beamer, …

다른 언어의 도움: LuaLaTeX, HaTeX, PyTeX, …

MathLetter.sty

MathLetter.sty

디자인을 하나하나 만들고 있기는 힘드니까...

- https://github.com/msquare-kaist/mathletter-package
- 매뉴얼: https://github.com/msquare-kaist/mathletter-package/blob/master/documents/manual.pdf
- 매뉴얼에 안 쓰여 있는 것: \PrintBibliography와 \Footnote
  - \PrintBibliography: .tex 파일 상단 `\documentclass`, `\usepackage` 밑에 참고문헌 .bib 파일을 추가하고 (`\addbibresource{filename.bib}`) `\end{document}` 전에 `\PrintBibliography` 를 입력하면 참고문헌이 나옵니다. (컴파일은 두 번 해야합니다.)
  - \Footnote: 그냥 `\footnote` 와 사용법은 같습니다. 여러 environment 안에 각주가 갇히는 현상을 방지하기 위한 매크로입니다.
- \MSquare: `\MSquare[1/2]`, `\MSquare`, `\MSquare[2]`
  - $\mathscr{M}^{2}$, $\mathscr{M}^{2}$, $\mathscr{M}^{2}$
  - 아직 소수는 입력이 안 되는데 언젠간 누군가 고칠 예정입니다.

끝

Appendix: More Advanced Usage

조금 더 복잡한 예시

이 정도는 익혀두면 과제할 때 손으로 계산할 필요가 없습니다

- ★ 유클리드 알고리듬을 구현해봅시다! 나누어질 수(#1)와 나눌 수(#2)를 받아 표를 만듭니다. 편의를 위해 #1 $\geq$ #2 > 0이라고 가정합니다.

조금 더 복잡한 예시

이 정도는 익혀두면 과제할 때 손으로 계산할 필요가 없습니다

- ★ 유클리드 알고리듬을 구현해봅시다! 나누어질 수(#1)와 나눌 수(#2)를 받아 표를 만듭니다. 편의를 위해 #1 $\geq$ #2 $>$ 0이라고 가정합니다.
- ★ 먼저 전체를 감쌀 array가 필요하므로, array로 감싸고 그 안에 있는 내용은 다른 매크로에게 맡깁니다.

```
\makeatletter\ExplSyntaxOn
\begin{array}{r@{\;=\;}r@{\;\cdot\;}l@{}l
                @{}l @{\quad}|@{\quad} r@{}r@{}r@{}r@{}r}
  ... % 다른 매크로
\end{array}
\makeatother\ExplSyntaxOff
```

조금 더 복잡한 예시

이 정도는 익혀두면 과제할 때 손으로 계산할 필요가 없습니다

- 유클리드 알고리듬을 구현해봅시다! 나누어질 수(#1)와 나눌 수(#2)를 받아 표를 만듭니다. 편의를 위해 #1 $\geq$ #2 > 0이라고 가정합니다.
- 먼저 전체를 감쌀 array가 필요하므로, array로 감싸고 그 안에 있는 내용은 다른 매크로에게 맡깁니다.

```
\makeatletter\ExplSyntaxOn
\begin{array}{r@{\;=\;}r@{\;\cdot\;}l@{}l
                @{}l @{\quad}|@{\quad} r@{}r@{}r@{}r@{}r}
  ... % 다른 매크로
\end{array}
\makeatother\ExplSyntaxOff
```

- 생각해 보면 나누어질 수와 나눌 수, 몫, 나머지, 이전 두 단계의 결과를 저장해야 되니까 10 개의 argument를 사용해야 하는데, 이것은 불가능합니다. 따라서 나누어질 수와 나눌 수는 전역변수로 저장하도록 합시다.

```
\gdef\@@@@a{#1}\gdef\@@@@b{#2}
```

조금 더 복잡한 예시

이 정도는 익혀두면 과제할 때 손으로 계산할 필요가 없습니다

- 그리고 안에 들어갈 매크로를 구현합니다.

```
...\begin{array}{r@{\;=\;}r@{\;\cdot\;}l@{}l
                @{}l @{\quad}|@{\quad} r@{}r@{}r@{}r@{}r}
  \euclid@lgorithm                    % initialize
  {#1}{#2}                            % a and b
  {\int_div_truncate:nn{#1}{#2}} % quotient q = floor(a / b)
  {\int_mod:nn{#1}{#2}}               % remainder r = a mod b
  {0}{1}                              % b = 0 * a + 1 * b
  {1}{\the\numexpr
      -\int_div_truncate:nn{#1}{#2}
                                \relax} % r = 1 * a + (-q) * b
\end{array}...
```

조금 더 복잡한 예시

이 정도는 익혀두면 과제할 때 손으로 계산할 필요가 없습니다

- 이제 안쪽에 들어갈 매크로를 만듭니다. 나머지가 0일 때와 아닐 때로 나눕시다.
- 나머지가 0일 때

```
\newcommand{\euclid@lgorithm}[8]{
  \int_compare:nNnTF {#4} = {0}
  {
    % if remainder is 0, terminate the recursion
    #1 & \mathbf{#2} & #3 &&&
  }
  { ... }
}
```

조금 더 복잡한 예시

이 정도는 익혀두면 과제할 때 손으로 계산할 필요가 없습니다

- ★ 나머지가 0이 아닐 때

```
\newcommand{\euclid@lgorithm}[8]{
  \int_compare:nNnTF {#4} = {0} { ... } {
    % print a line
    #1 & #2 & #3 \;&\;+\;&\; #4 & #4 &\;=\;&\;\a \;\cdot\; &
    \,{\ifnum#7<\z@ (#7) \else #7\hphantom{)}\fi} + \b \;\cdot\;&
    \,{\ifnum#8<\z@ (#8) \else #8\hphantom{)}\fi} \\%
    % call itself recursively
    \euclid@lgorithm
    {#2}{#4}
    {\int_div_truncate:nn{#2}{#4}}
    {\int_mod:nn{#2}{#4}}
    {#7}{#8}
    {\the\numexpr #5 - \int_div_truncate:nn{#2}{#4} * #7\relax}
    {\the\numexpr #6 - \int_div_truncate:nn{#2}{#4} * #8\relax}%
  }
}
```

배열을 다루는 법

plain TEX에서는 (첫 번째 방법)

- ★ Expand가 되지 않는 토큰을 넣어서 delimiter의 역할을 하게 합니다.
- ★ 예를 들어서 여러 식이 합동인 상황 $F_1 \equiv F_2 \equiv F_3 \pmod{N}$을 나타내고 싶을 때 `\eqv{F_1}{F_2}{F_3}\mod{N}` 이라고 쓰면 편할 것입니다. 이런 상황은 다음과 같이 구현할 수 있습니다.

```
\makeatletter
\def\eqv#1#2{\let\mod=\m@d{#1}\expandafter\eqvB{#2}}
\def\eqvB#1{
  \@ifnextchar\mod{\equiv#1}{\equiv#1\expandafter\eqvB}
}
\def\m@d#1{\quad({\operatorname{mod}}\;#1)}
\makeatother
```

배열을 다루는 법

plain TEX에서는 (두 번째 방법)

- ★ 매크로 정의에 delimiter를 명시적으로 넣습니다. (패턴 매칭 기능)
- ★ 토큰 리스트를 뒤집는 매크로를 만들어봅시다.

```
\def\firstoftwo#1#2{#1}
\def\secondoftwo#1#2{#2}

\def\rev#1#2\revA#3\revB{%
  \if\relax\detokenize{#2}\relax
    \expandafter\firstoftwo
  \else
    \expandafter\secondoftwo
  \fi{#1#3}{\rev#2\revA#1#3\revB}%
}
```

배열을 다루는 법

plain TEX에서는 (세 번째 방법)

- ★ 배열 원소를 각각 다른 매크로로 정의합니다. `\csname NAME \endcsname`은 `\NAME`과 같은데, NAME이 알파벳만으로만 이루어져 있지 않아도 정의됩니다. (아래 예제에서는 카운터를 LATEX 카운터를 썼습니다.)

```
\makeatletter
\expandafter\def\csname my@array@1\endcsname{1}
\expandafter\def\csname my@array@2\endcsname{2}

% with counter
\newcounter{arrayindex}
\setcounter{arrayindex}{3}
\expandafter\def\csname my@array@\value{arrayindex}\endcsname
  {\value{arrayindex}}
\makeatother
```

배열을 다루는 법

LaTeX 3에서는

- LaTeX 3에는 토큰 리스트를 위한 함수들이 따로 마련되어 있습니다. `tl` 이 포함된 매크로들이 그것입니다.
- 그런데 제가 어떻게 쓰는지 몰라서...
- `https://www.texdev.net/2011/12/26/programming-latex3-token-list-variables/`

진짜 끝