5. Создание новых команд

LaTeX в Вышке

2 мая 2022 г.

Официально новые команды называются макроопределениями, а в разговорной речи — макросами.

1 Команды без аргументов

Часто писать один и тот же длинный набор команд утомительно. Потому для упрощения жизни обычно создают новые команды.

Например, если в тексте часто встречается символ «$\stackrel{\text{def}}{=}$», можно определить в преамбуле новую команду «`\eqdef`»:

```
\newcommand{\eqdef}{\stackrel{\mathrm{def}}{=}}.
```

И теперь для печати этого символа достаточно написать `$\eqdef$`.

Новая команда TeX'а, которую мы определили, называется макросом. Рассмотрим точные правила для создания макросов средствами LaTeX'а.

Для создания макросов используется (обычно в преамбуле) конструкция

```
\newcommand{\имя-команды}{замещаемый-текст}
```

Первый аргумент — имя, которое вы придумали для вашего макроса. Имена макросов должны подчиняться тем же правилам, что имена TeX'овских команд: либо backslash и после него одна не-буква, либо backslash и после него — последовательность букв.

Второй обязательный аргумент команды `\newcommand`, называемый «замещающим текстом», сообщает TeX'у смысл макроса: на этот текст ваш макрос будет замещаться в процессе трансляции (как говорят, макрос будет «разворачиваться»). Сюда можно писать любой набор команд и текста.

При пользовании командой `\newcommand` нельзя в качестве имени макроса выбирать имя уже существующей команды или окружения (если вы попробуете так сделать, LaTeX выдаст сообщение об ошибке).

Во втором аргументе команды `\newcommand` (иными словами, в «замещающем тексте») вместе с каждой открывающей фигурной скобкой должна присутствовать соответствующая ей закрывающая.

Совет: не используйте в новых командах знак $; если изменяете стиль текста, то делайте это внутри группы (т.е. в фигурных скобках: `{bf текст...}` или `\textbf{text}`).

Переопределить значение уже существующей команды с помощью `\newcommand` невозможно. Для такого рода целей используется команда `\renewcommand`. Она устроена точно так же, как `\newcommand`, с тем отличием, что в качестве ее первого аргумента надо указывать имя уже *существующей* команды.

2 Команды с аргументами

Для создания новых команд с аргументами используют общую форму команды `\newcommand`:

```
\newcommand{\имя-команды}[кол-во-аргументов][по-умолчанию]{текст-подстановки}
```

Если указано значение *по умолчанию*, то **первый** аргумент создаваемой команды становится необязательным и в случае пропуска ему присваивается значение по умолчанию. Для ссылки на первый аргумент в тексте подстановки следует написать `#1`, на второй — `#2` и так далее.

Например, определение новых команд

```
\newcommand{\smb}[2]{\left(\frac{#1}{#2}\right)}
\newcommand{\xvec}[2][n]{ (#2_1,#2_2,\ldots,#2_#1) }
```

позволяют, например, написать

$$\left(\frac{a+b}{c}\right),\ X=(x_1,x_2,\ldots,x_n),\ Y=(y_1,y_2,\ldots,y_n),\ Z=(z_1,z_2,\ldots,z_k)$$

3 Немного о счетчиках

В системе LATEX можно определять целочисленные переменные, называемые счётчиками. Такие переменные в основном используются для хранения номеров структурных частей документа, как, например, главы, разделы, номера страниц. Иногда счётчики выступают в роли целочисленных параметров. Имя счётчика состоит из последовательности латинских букв без символа обратной косой черты. Системой изначально определены счётчики

`part, chapter, section, subsection, subsubsection, paragraph, subparagraph,`

отвечающие за нумерацию соответствующих структурных элементов документа. Счетчики

`page, equation и footnote`

хранят соответственно текущие номера страницы, формулы и сноски. Определены также некоторые другие стандартные счётчики.

Команда `\newcounter{новый-счетчик}[главный-счётчик]` ссоздаёт новый счётчик и инициализирует его нулевым значением. Если счётчик с таким именем уже существует, печатается сообщение об ошибке. Если указан главный счётчик, то создаваемый новый счётчик становится ему подчинённым. Это означает, что при увеличении главного счётчика на единицу рассматриваемыми ниже командами новый счётчик сбрасывается в нулевое значение. Такой механизм позволяет, например, получит вложенную нумерацию разделов документа (например, в начале каждой новой главы нумерация разделов начинается заново).

Для изменения значения счётчика предназначены команды

```
\setcounter{счётчик}{значение}
\addtocounter{счётчик}{значение}
```

Первая присваивает, а вторая добавляет указанное значение к счётчику.

Значение счётчика возвращает команда `\value{счётчик}`, которая может быть использована в тех местах, где системе требуются целочисленные данные. Например, в аргументе значение команд `\setcounter` и `\addtocounter`.

Вывести значение счётчика на печать можно командами

`\arabic{счётчик}, \roman{счётчик}, \Roman{счётчик}, \alph{счётчик}, \Alph{счётчик}.`

Первые три печатают значение счётчика арабскими, строчными римскими и заглавными римскими цифрами. Последние две печатают идущую под номером, равным значению счётчика, строчную латинскую и прописную латинскую букву.

Команда `\newcounter{счётчик}` не только создаёт новый счётчик, но и определяет команду `\the-`*счётчик*, задающую его визуальное представление. По умолчанию последняя печатает значение счётчика арабскими цифрами, но может перекрываться посредством `\renewcommand`. Так и сделано в классе book при определении счётчиков, отвечающих за номера разделов. Вот некоторые из них:

```
\newcounter{part}
\renewcommand{\thepart}{\Roman{part}}
\newcounter{chapter}
\renewcommand{\thechapter}{\arabic{chapter}}
\newcounter{section}[chapter]
\renewcommand{\thesection}{\thechapter.\arabic{section}}
\newcounter{subsection}[section]
\renewcommand{\thesubsection}{\thesection.\arabic{subsection}}
```

На практике значения счётчиков чаще всего приходится увеличивать на 1. Для этого предусмотрены две специальные команды:

```
\stepcounter{счётчик}, \refstepcounter{счётчик}.
```

Кроме своего основного действия они сбрасывают значения всех счётчиков, подчинённых указанному счётчику. Команда `\refstepcounter` к тому же задаёт значение текущей метки, запоминаемой `\label`, равным `\the-`*счётчик*, и потому используется в определении всех стандартных нумерованных объектов.

4 Создание новых окружений

4.1 Общий случай

В тех случаях, когда для достижения необходимого нам эффекта требуется сложная последовательность команд в начале и в конце какого-то текста, LaTeX дает возможность оформить соответствующие макросы в виде нового окружения. Как это делается, разберем на примере.

Предположим, нам хочется взять в рамку абзац текста шириной 5. Один из возможных способов таков:

```
\begin{tabular}{|p{5cm}|}
\hline
Этот текст будет заключен в рамку.
Как видите, окружение,
предназначенное для верстки таблиц,
можно использовать и для этих целей.\\
\hline
\end{tabular}
```

Этот текст будет заключен в рамку. Как видите, окружение, предназначенное для верстки таблиц, можно использовать и для этих целей.

Если таких рамок с текстом у вас много, то можно сократить число нажатий на клавиши, определив окружение с именем, скажем, ramka, так, чтоб можно было бы просто писать

```
\begin{ramka} Этот текст будет ... этих целей. \end{ramka}
```

Определяется окружение так:

```
\newenvironment{ramka}{\begin{tabular}{|p{7cm}|}\hline}{\\\hline\end{tabular}}
```

В общем случае команда `\newenvironment` имеет такой формат:

```
\newenvironment{имя}{открывающие_команды}{закрывающие_команды}
```

Здесь *имя* — имя определяемого окружения, *открывающие_команды* — команды и/или текст, подставляемые вместо команды `\begin` с именем окружения, *закрывающие_команды* — команды и/или текст, подставляемые вместо команды `\end` с именем окружения.

Нелишне напомнить, что команды `\begin` и `\end`, ограничивающие окружение, ограничивают группу: все неглобальные определения и изменения параметров, происходящие внутри окружения, забываются по выходе из него.

Новые окружения можно определять так, чтобы они принимали аргументы. Общая схема такова:

```
\newenvironment{имя}[к-во][по-умолчанию]{начало}{конец}
```

Здесь использовать аргументы можно **только** в начале окружения (т.е. во втором обязательном аргументе). В остальном синтаксис аналогичен команде `\newcommand`. Для переопределения старых окружений используют с тем же синтаксисом команду `\renewenvironment`.

4.2 Теоремоподобные окружения

Окружения, используемые в LATEX'е для оформления фрагментов текста типа «теорема», заранее не определены. Дело в том, что количество различных типов объектов наподобие теоремы, присутствующих в одном тексте, может быть достаточно велико (предложение, утверждение, лемма, определение, замечание,...), так что LaTeXв целях экономии машинной памяти и исходя из того, что на все вкусы таких окружений все равно не напасешься, определять их предоставляет вам. Объявить нужное вам утверждение можно, поместив в преамбулу специальную команду

```
\newtheorem{имя}{заголовок}
```

Вариант этой команды со звездочкой порождает ненумерованное утверждение. Вместе с окружением типа «теорема» автоматически создается и счетчик, хранящий его номер. Имя этого счетчика совпадает с именем окружения; чтобы изменить представление на печати номеров наших «теорем», можно обычным образом переопределить соответствующую `the`-команду.

Пусть в нашем тексте присутствуют «предложения». Давайте создадим окружение predl таким образом, чтобы можно было, например, писать

Предложение 1 (Пифагор)
Пифагоровы штаны на все стороны равны.

```
\begin{predl}[Пифагор]
Пифагоровы штаны на
все стороны равны.
\end{predl}
```

Для создания такого окружения используется команда `\newtheorem`:

```
\newtheorem{predl}{Предложение}
```

Создавая утверждение, можно подчинить его нумерацию нумерации другого (главного) объекта. Для этих целей у команды `\newtheorem` предусмотрен необязательный аргумент `главный-счётчик`

```
\newtheorem{имя-окружения}{заголовок}[главный-счетчик]
```

Например, если мы захотим подчинить нумерацию предложений счетчику `\subsection`, то в преамбуле напишем

```
\newtheorem{predl}{Предложение}[subsection]
```

После этого можно будет писать, например, вот что:

Предложение 4.2.1 (Пифагор)
Пифагоровы штаны на все стороны равны.

```
\begin{predl}[Пифагор]
Пифагоровы штаны на
все стороны равны.
\end{predl}
```

По умолчанию каждый тип из созданных вами утверждений будет нумероваться самостоятельно. Однако, объявляя новое утверждение, можно не вводить для него свой счётчик, а привязать к какому-либо существующему, что приведёт к совместной нумерации утверждений (например, совместно нумеровать предложение и теоремы). Указать этот определённый ранее счётчик можно в ещё одном необязательном аргументе команды `\newtheorem`:

```
\newtheorem{имя}[счётчик]{заголовок}
```

4.3 Стиль теоремоподобных окружений

Американское математическое сообщество (AMS) предоставляет специальный пакет `amsthm`, позволяющий быстро и качественно создавать их по всем правилам искусства. Этот пакет должен загружаться после `amsmath`.

Пакет `amsthm` позволяет задавать стиль утверждений, влияющий на их внешний вид. Изначально определены следующие три стиля: `plain` (обычный; для теорем) `definition` (для определений) `remark` (для замечаний и примеров).

Текущий стиль устанавливается командой `\theoremstyle{стиль}`, а используют его все последующие команды `\newtheorem`.

В преамбуле это выглядит примерно так:

```
\theoremstyle{plain}
\newtheorem{theorem}{Теорема}[chapter]
\newtheorem{cor}[theorem]{Следствие}
\theoremstyle{definition}
\newtheorem{definition}{Определение}[chapter]
\theoremstyle{remark}
\newtheorem*{remark}{Замечание}
```

Для доказательств пакет `amsthm` предусматривает специальное окружение `\begin{proof}[имя]`. Если задано имя, то оно используется в качестве заголовка доказательства. В противном случае печатается слово «Доказательство». Если доказательство заканчивается *формулой*, то завершите ее командой `\qedhere` (нужно для того, чтобы знак завершения доказательства попал на ту же строку).