Степанов Алексей Владимирович

ЯЗЫКИ РАЗМЕТКИ

ВВЕДЕНИЕ

Мы начинаем серию статей о языках разметки текста. Эта серия является отредактированными записями лекций, прочитанных автором в Санкт-Петербургском электротехническом университете осенью 2007 года. В связи с этим я хочу от всей души поблагодарить Сергея Николаевича Позднякова за организацию лекций, Сергея Георгиевича Иванова за предоставленные записи, а также всех слушателей, без которых этот текст не появился бы на свет.

Что такое язык разметки? Строго говоря, любой формальный язык можно назвать языком разметки. Отличается не синтаксис, а семантика языка: *основное назначение языков разметки – форматирование текста.* Хотя, как мы увидим впоследствии, у языков разметки существуют и другие функции.

Каких языков касается курс? Я выбрал три из них. Во-первых, TeX, точнее, его модификация, LaTeX, который я знаю хорошо, и который является очень мощным средством форматирования *печатных* текстов. Я буду много рассказывать именно про этот язык.

Второй язык – XML – расширяемый язык разметки, обладающий массой достоинств. Он очень простой и очень универсальный. У него всего один недостаток. Он замечательно размечает текст, но никто не знает, что после этого с разметкой делать. Можно написать программу, которая интерпретирует эту разметку, можно сделать формальное преобразование одного XML-документа в другой, но нельзя написать заранее универсальную программу, интерпретирующую XML разметку. Фактически любой язык разметки является конкретизацией XML, возможно, после некоторого формального преобразования. Про XML я буду рассказывать только идеи без всякой конкретики.

Третий язык, который будет затронут, – это конкретизация XML для верстки Интернет-страниц – HTML[1] , в переводе – язык разметки гипертекста. Он немного сложнее, чем XML, просто потому, что, изучая его, надо запоминать конкретные названия форматирующих элементов и их атрибуты. Про него я также буду рассказывать немного.

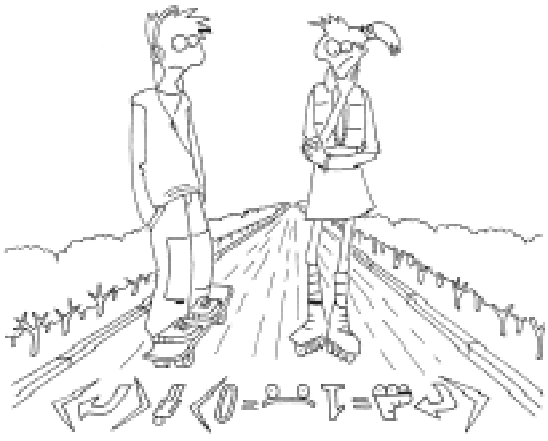
Несколько слов о том, как я дошёл до жизни такой: я, математик, занимающийся наукой, стал интересоваться компьютерными тонкостями, притом что большая часть математиков использует компьютер исключительно как пишущую машинку.

Языками разметки я заинтересовался, после того как узнал, что бывает локальная разметка, а бывает разметка логическая. Когда я понял прелесть логической разметки, мне захотелось это всё узнать поближе. Начинал я с TeX, потому что математические журналы принимают только текст, набранный в TeX. Пришлось научиться основам компьютерной верстки. А однажды заведующий нашей кафедры поручил мне

[1] Не XHTML, потому что на том уровне, на котором это будет затронуто, расширяемость не играет роли.

составить подборку задач, различающихся только числовыми данными (которые были уже сделаны), чтобы каждому студенту выдать готовый распечатанный вариант с полным текстом. Поскольку книгу Кнута «Все про TeX» я прочитал незадолго до этого, то за час мне удалось справиться с поставленной задачей. Меня самого поразило, как легко получилось сделать вставку цифр в текст. Конечно, не сразу все вопросы были решены удовлетворительно. Например, сначала печаталось $x + -1y$ вместо $x - y$. То есть число –1 в формулу вставлялось в том формате, в каком его создала программа. Из этой идеи – заставить TeX вставлять числа в готовые шаблоны, подбирая на ходу правильный формат, родился электронный задачник, активно используемый на кафедре «Высшей математики № 2» Санкт-Петербургского электротехнического университета уже в течение десяти лет. Так мне пришлось поближе узнать TeX уже в качестве своеобразного языка программирования.

В отличие от HTML, который интерпретируется браузером «online», TeX является компилируемым языком разметки. После компиляции создается файл формата, приближающегося к языку принтера: DVI[1], PDF[2] или PostScript. Единственным широко распространенным из них является PDF, но небольшие тексты в этом формате превращаются в непропорционально



...сами маропоследовательности выглядят следующим образом...

большие файлы, потому что файлы содержат в себе все использованные шрифты. И тут я прочитал, что есть язык XML и что LaTeX и XML идейно очень похожи и легко переводятся друг в друга. С другой стороны, уже появляются браузеры, умеющие интерпретировать XML разметку на основании таблиц стилей. Тогда я понял, что для экономии размеров файлов хорошо бы переводить LaTeX в XML. К сожалению, в настоящий момент для текстов, содержащих достаточно много формул, экономии не получится из-за отсутствия то ли языка MathML[3], то ли из-за того, что он не поддерживается стандартными браузерами.

Итак, что такое языки разметки? Язык разметки – это множество макропоследовательностей и непечатных символов. Текст на таком языке обычно выглядит следующим образом: это текст для печати, перемежаемый макропоследовательностями и непечатными символами, которые указывают компьютеру, как он должен форматировать этот текст.

Примеры

В языке XML непечатными символами являются треугольные скобки, которые используются только для создания макропоследовательностей, а сами маропоследовательности выглядят следующим образом:

<идентификатор атрибуты> или **</идентификатор>,**

которые называются *открывающим и закрывающим тэгом* соответственно. Атрибуты – это последовательность равенств вида

название_атрибута = значение_атрибута,

идущих через запятую. Идентификатор является именем тэга и обычно указывает, по какому образцу нужно форматировать текст, находящийся между открывающим и закрывающим тэгом: как заголовок, как абзац, как таблицу и т. п., в то время как атрибуты указывают точные параметры соответствующего объекта. В XML допускается писать

[1] Device Independent, в переводе – независимый от устройства (правда зависимый от наличия шрифтов).
[2] Portable Document Format, в переводе – мобильный формат документов.
[3] Mathematical Markup Language, в переводе – математический язык разметки.

<идентификатор атрибуты/>

вместо

<идентификатор атрибуты></идентификатор>.

Есть ещё правило, чтобы эти группы были правильно вложены, то есть последовательность **<a>......** недопустима.

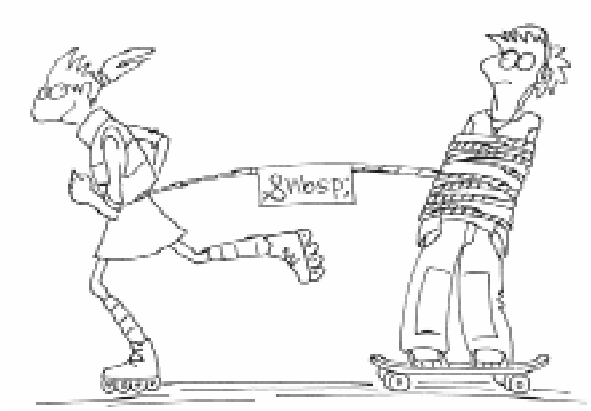
В HTML допускаются еще макропоследовательности вида

&идентификатор;

и, соответственно, символ **&** становится непечатным. Кроме того, HTML допускает наличие непарных тэгов, хотя последовательность **<a>.........** по-прежнему недопустима. Но основное отличие HTML от XML состоит в том, что в языке HTML набор имен тэгов конечен, в то время как в XML каждый может придумать новое имя тэга. Это и означает, что XML – расширяемый язык.

В HTML у каждой макропоследовательности есть свое предопределенное значение. Кое-что можно переопределить в таблице стилей, но тэг **<I>** не будет начинать новый абзац, а с помощью тэга **<P>** невозможно отформатировать таблицу. Макропоследовательности, начинающиеся с **&**, вообще невозможно переопределить. Например, ** **; всегда означает неразрывный пробел, а **&Phi**; – греческую букву Ф.

В языке TeX макропоследовательность состоит из символа \ и последовательности букв. Макропоследовательность прерывается, когда встречается не буква, то есть либо пробел, либо цифра, либо знак препинания. Другой вид макропоследовательности – **\?**, где, вместо знака вопроса, стоит любой знак препинания. Есть еще и третий сорт макропоследовательностей – **^^X**, где вместо **X** может стоять любая буква латинского алфавита. Эти макропоследовательности используются для обозначения символов, которые нельзя ввести с клавиатуры в текстовый файл, а также символа перевода строки. Макрос **^^X** обозначает символ, ASCII-код которого равен порядковому номеру соответствующей буквы в алфавите. Например, **^^M** обозначает перевод строки (символ с кодом 13), **a^^a** – символ ESC. Кроме \, непечатными символами являются также **& # $ % ^ _ { }** и **~**.



* всегда означает неразрывный пробел ...*

Деление символов на печатные и непечатные весьма условно. Вообще, в языке разметки каждый символ относится к какой-нибудь *категории символов*. В TeX эта идея формализована, то есть все символы разделены на 16 категорий (см. табл. 1).

Строго говоря, TeX имеет дело не с символами, а с токенами. *Токен* – это символ с учетом кода категории, которую символ имеет на момент считывания, или макропоследовательность. Преобразования символов и макропоследовательностей в токены происходит на этапе претрансляции, как и некоторые другие операции.

ПОЧЕМУ LATEX?

LaTeX – язык высокого уровня, писать на нём удобно и просто. Если вы не хотите ничего форматировать, можете просто написать текст без всяких макропоследовательностей, и он уже отформатируется довольно прилично, по крайней мере, лучше, чем это сделает MSWord[1]. Создатель языка TeX Дональд Кнут начинает свою книгу «Все про TeX» с того, как TeX форматирует абзацы и страницы. MSWord реально форматирует строку. От того, что написано в следующей строке, форматирование текста в данной строке не зависит (по край-

[1] Никакой антирекламы! Я сравниваю с MSWord только потому, что этот редактор фактически стал единственным, которым пользуются непрофессионалы.

ней мере, у меня складывается такое ощущение). Если MSWord форматирует абзац по ширине, а в строке находится неразрывная последовательность символов (формула или Интернет-адрес), то остаются большие пробелы. TeX при этом постарается переформатировать предыдущие строки, вытолкнув или втянув слово, мешающее форматированию проблемной строки. Если же это не даст удовлетворительных результатов, то TeX, по крайней мере, «скажет» вам, что абзац отформатирован плохо (что такое «плохо» определяется некоторыми параметрами, значения которых можно менять в любой момент). Еще лучше преимущества TeX видны при разбиении текста на страницы.

Другим достоинством TeX является *скорость набора*. Ясно, что ввести с клавиатуры **\alpha** намного быстрее, чем найти символ α в таблице символов. Более того, если эта греческая буква встречается очень часто, то, написав в преамбуле TeX-файла

```
\def\a{\alpha},
```

далее можно писать **\a** для печати этого символа. То же самое происходит с переключением шрифтов и другими «удобствами» MSWord – все нужно делать мышью, что резко снижает скорость набора. Утверждение о том, что для набора в языке TeX необходимо много знать, также не выдерживает никакой критики. TeX достаточно хорошо документирован, поэтому для набора текста с обычными формулами достаточно иметь под рукой таблицы символов, которые вскоре оказываются ненужными, потому что *макропоследовательности TeX легко запоминаются.* Во многих случаях вы даже не будете заглядывать в эти таблицы. Угадайте, например, как в TeX называется символ β[1] ? Если же вам понадобятся сложные средства форматирования, такие как нестандартные таблицы или списки, плавающие иллюстрации (то есть иллюстрации, место вставки которых находит программа) и т. п., то от необходимости учиться вас не избавит ни один редактор.

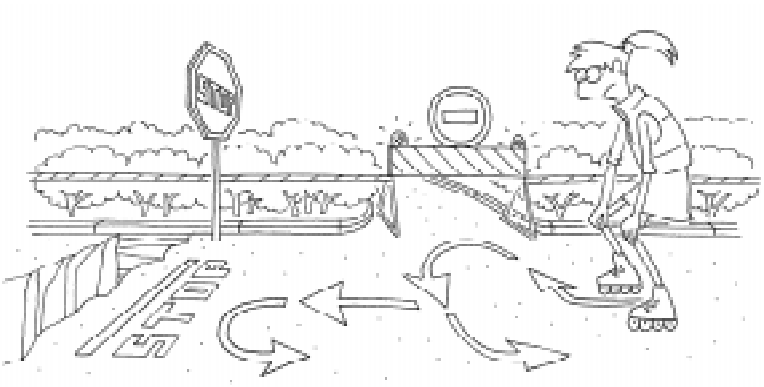
Однако основным преимуществом LaTeX перед MSWord является наличие *логической разметки и возможность переформатировать текст*, не изменяя его.

Таблица 1

Код	Назначение	Пример
0	Оператор управляющей последовательности	`\`
1	Оператор начала группы	`{`
2	Оператор конца группы	`}`
3	Переключатель математического режима	`$`
4	Метка табуляции	`&`
5	Конец строки	`^^M`
6	Оператор макропараметра	`#`
7	Оператор надстрочного индекса	`^`
8	Оператор подстрочного индекса	`_`
9	Игнорируемый символ	
10	Символ пробела	`пробел, TAB`
11	Буква	`A..Z, a..z`
12	Другой символ	`@, знаки препинания`
13	Активный символ	`~, любая макропоследовательность`
14	Оператор комментария	`%`
15	Неверный символ	`^^a`

[1] Надеюсь, что вам не удалось не угадать. Конечно **\beta** !

Более подробно эта идея, присущая всем языкам разметки, будет обсуждаться в следующих разделах. Эта черта в гораздо меньшей степени присуща базовой версии TeX – Plain TeX. Именно поэтому я буду говорить только про LaTeX.



...следуя указаниям логической разметки... все мыслимые и немыслимые локальные инструкции...

Кроме стандартных способов преобразования текста, размеченного LaTeX, в файл для печати, существует транслятор, который переводит из TeX в HTML. Везде, где нет математических формул, этот транслятор действует хорошо и создаёт довольно экономный HTML-код. В то же время, HTML-код, созданный MSWord, больше в несколько раз за счёт наличия массы никому не нужной (и даже вредной) локальной разметки. Возникает ощущение, что MSWord не доверяет ничему другому разметить текст, следуя указаниям логической разметки, методично вставляя все мыслимые и немыслимые локальные инструкциио том, как должен выглядеть этот текст.

ПЕРВАЯ ПРОБА ПЕРА В LATEX И HTML

Сначала приведем различия между различными версиями TeX, чтобы вы не путались, встречая слова emTeX, MikTeX, AmSTeX, LaTeX и т. п. Язык TeX состоит из набора нераскрываемых макропоследовательностей, которые интерпретируются программой tex.exe. Являясь расширяемым языком, TeX позволяет пользователю определить свои собственные макропоследовательности, написав макроопределения, то есть что-нибудь типа `\def\x{...}`. Набор основных раскрываемых макроопределений называется форматом. Широко известны четыре различных формата: Plain TeX, AmSTeX, LaTeX2.09 и LaTeX2e. Слова emTeX, MikTeX и VTeX (векторный TeX) относятся не к языку TeX, а к оболочке, то есть набору программ, перерабатывающих TeX-файл в DVI или другие форматы, просматривающие DVI-файлы, и набору предоставляемых шрифтов и т. п.

Логическая разметка появилась в форматах AmSTeX и LaTeX2.09. Так как LaTeX2e объединяет достоинства этих форматов, мы будем «разговаривать» только на этом языке. В связи с этим в LaTeX появились *окружения,* то есть парные макросы

`\begin{идентификатор}...\end{идентификатор},`

аналогичные тэгам языка XML.

Попробуем теперь написать простенький текст на языке LaTeX, сделать из него DVI-файл и посмотреть на экране, что получилось[1]. Первой строкой LaTeX-файла должно быть указание *класса документа* и его параметров, которые называются *стилевые опции*. После преамбулы (в просторечии – шапки) документа, которая может быть пустой, должно начинаться окружение document. Внутри этого окружения должен находиться собственно размеченный текст. Итак, при помощи текстового редактора наподобие блокнота пишем в файл example1.tex следующие строки.

```
\documentclass[12pt,a4paper]{article}
\usepackage[Russian]{Babel}
\begin{document}
Сегодня наш первый урок по языку
\LaTeX. \par Нам очень {\bfseries
скучно}.
\end{document}
```

[1] Предполагается, что на вашем компьютере установлена какая-то оболочка TeX, содержащая формат LaTeX2e.

Это была моя первая проба пера в LaTeX
В командной строке пишем

```
latex example1
```

после чего получаем файл **example1.dvi**. Кликаем его мышью и (если у вас в «свойствах папки» выставлена правильная ассоциация с расширением DVI) получаем на экране следующую картинку.

Сегодня наш первый урок по языку LaTeX.
Нам очень **скучно**.

Давайте теперь разберем, что же мы написали, и почему это превратилось именно в такую картинку. Основными классами документов LaTeX являются **letter**, **article**, **book** и **proc**, а также их аналоги по стандартам Американского Математического Общества **amsart**, **amsbook** и **amsproc**. В квадратных скобках написаны стилевые опции: размер шрифта и размер бумаги. Команда **\usepackage** говорит о том, что надо использовать дополнительные макроопределения из *стилевого пакета.* В данном случае стилевой пакет **Babel** предоставляет возможности набора в многоязычной среде. Со стилевой опцией **Russian** он дает возможности пользоваться кириллицей, а также устанавливает русские стандартные названия дат, списка литературы, имен таблиц и рисунков и многое другое. Текст внутри окружения **document** отобразился почти один к одному. Макропоследовательность **\LaTeX** – это логотип LaTeX. Макропоследовательность **\par** – сокращение от слова «paragraph», что в переводе означает «абзац», дает указание начать новый абзац. Слово «скучно» напечатано полужирным шрифтом, потому что оно находится в группе (в фигурных скобках), внутри которой действует шрифтовая команда **\bfseries** (**bf** – сокращение от bold face). После закрытия окружения **document** может присутствовать любой текст, но на печать он выведен не будет.

А теперь давайте попробуем получить аналогичную картинку с помощью языка HTML. В файле **example1.htm** при помощи текстового редактора пишем следующий текст.

```
<html>
<head>
<title> Первая проба пера в HTML </title>
</head>
<body>
<p>
Сегодня наш первый урок по языку HTML.
</p>
<p>
Нам очень <b>скучно</b>.
</p>
</body>
<html>
```

После этого кликаем мышью на **example1.htm** и в окне браузера получаем картинку:

Сегодня наш первый урок по языку HTML.

Нам очень **скучно**.

Почему получилось именно это? Тэг **<html>** является обязательным для указания браузеру, с каким языком он имеет дело. Внутри тэга **<head>** пишется преамбула (шапка) документа, содержащая информацию не для печати. Например, там может присутствовать тэг **<title>**, чей текст отображается в заголовке окна браузера. Тэг **<body>** является аналогом окружения **document**. Тэг ** ** означает абзац. Обратите внимание, что, в отличие от LaTeX, содержимое абзаца находится внутри тэга. В LaTeX это редкое отступление от принципа логической разметки в пользу краткости. Справедливости ради надо сказать, что и в HTML современные браузеры сами вставят закрывающий тэг ** ** перед началом следующего абзаца, если вы его пропустите. Тэг **<b>** означает переход на полужирный шрифт. Я намеренно избежал логотипа LaTeX. В HTML также как и в LaTeX, визуальные элементы (в частности буквы) можно двигать в произвольном направлении с помощью стилей, однако это выходит за рамки первоначального знакомства с языком.

МАКРОПОСЛЕДОВАТЕЛЬНОСТИ LATEX

С макропоследовательностями языка HTML все ясно, а вот про LaTeX надо ска-

зать еще несколько слов. Макропоследовательности LaTeX могут иметь обязательные и необязательные аргументы. Необязательный параметр (за редким исключением он один) ставится сразу после имени макроспоследовательности в квадратных скобках. Уже известным примером необязательного параметра макроспоследовательности **\usepackage** является список стилевых опций. Обязательные параметры являются либо токенами, либо группами, заключенными в фигурные скобки. Изредка в строке параметров должны присутствовать предопределенные токены, являющиеся разделителями параметров. Строка параметров макроспоследовательности должна быть известна. Например, макропоследовательность **\emph** имеет один параметр, который будет напечатан *выделенным* шрифтом (в общеупотребительных классах документов он будет напечатан курсивом, если основной шрифт прямой и прямым, если основной шрифт курсивный). Таким образом, чтобы получить фрагмент предыдущего предложения, в TeX-файле надо написать:

```
который будет напечатан \emph {выделенным} шрифтом
```

Если же пропустить фигурные скобки, то на печати получится текст, который будет напечатан *в*ыделенным шрифтом, потому что параметром макропоследовательности **\emph** будет только буква «в». При определении параметров макропоследовательности TeX игнорирует пробелы, если только они не являются обязательными разделителями параметров. Более подробно про то, как TeX обходится с пробелами, будет сказано в разделе «Претрансляция TeX».

Иногда создатели макропоследовательностей предусматривают для них вариант *со звездочкой*. Например, макропоследовательность **\section**, которая начинает новый параграф документа, имеет один обязательный аргумент – название параграфа. По умолчанию, это название на печати предваряется номером параграфа (LaTeX поддерживает автоматическую нумерацию и ссылки). В варианте со звездочкой номер не ставится.

Макропоследовательности LaTeX имеют разный семантический смысл. Условно их можно разделить на пять категорий.

1. Названия символов.

2. Макрокоманды, то есть макропоследовательности, которые говорят программе, как нужно обрабатывать их параметры или весь текст внутри группы.

3. Окружения, которые имеют вид

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

4. Регистры: счетчики, размеры, промежутки и др.

5. Операторы.

Для краткости макропоследовательности первой категории в дальнейшем будут называться макросами, а второй – командами.

Про различные виды макроопределений мы поговорим подробно в разделе «Программирование в LaTeX», а сейчас приведем простейшие конструкции.

```
\newcommand\xxx[n][умолчание]{определение}
\renewcommand\xxx[n][умолчание]{определение}
\def\xxx строка_параметров{определение}
```

Здесь **\xxx** – это определяемая макропоследовательность, **n** – количество параметров, **умолчание** – значение необязательного параметра по умолчанию, а **определение** – сам текст макроопределения, в котором формальные параметры обозначаются символами **#1,...,#n**. Если **[умолчание]** отсутствует, то параметров по умолчанию нет, а если присутствует, то необязательный параметр имеет номер 1. Отсутствие указания на количество параметров означает, что параметров нет вообще. В макроопределении **\def строка_параметров** состоит из символов **#1,...,#n** и, возможно, ограничителей параметров, то есть произвольного текста перед, между и после обозначений параметров. Если попытаться определить макропоследовательность, которая уже была определена ранее, то команда **\newcommand** выдаст сообщение об ошибке и ничего не переопределит, а остальные две команды сработают так, как будто никакого определения ранее не было. Зато **\renewcommand** выдаст сообщение об ошиб-

ке в случае, когда макропоследовательность не была определена ранее. Макроопределения обычно пишут в преамбуле TeX-файла, хотя не запрещается вставлять их в любое место документа.

Приведем несколько примеров.

```
\newcommand\myheading[2]
          [\normalfont\bfseries]{\par
 \medskip{#1#2.}\par\smallskip}
```

Мы определили команду для форматирования заголовка. Заголовок будет печататься отдельным абзацем, до него вставляется вертикальный промежуток среднего размера, а после него – маленький вертикальный промежуток. При этом, по умолчанию, заголовок печатается прямым полужирным шрифтом. Теперь в тексте документа можно написать

```
\myheading{первый заголовок}
```
или
```
\myheading[\itshape]{Первый заголовок}
```

Во втором случае заголовок будет напечатан курсивом.

```
\usepackage{amssymb}
\renewcommand\le{\leqslant}
```

Здесь мы переопределили символ ≤ на русский стандарт (нижняя черта – наклонная). Так как в предопределенном наборе символов LaTeX этого символа нет, то перед макроопределением мы подгрузили пакет **assymb** с дополнительными математическими символами.

```
\def\SkipSpace#1 #2{#1#2}
```

Обратите внимание на пробел, стоящий после **\SkipSpace#1**, в нем вся суть этого примера. Макрокоманда **\SkipSpace** убирает первый встретившийся после нее пробел. Например,

```
\SkipSpace A. B. Степанов
```

преобразуется в **A. B. Степанов**, потому что *пробелы непосредственно после макропоследовательности игнорируются* на этапе претрансляции. Таким образом, первый фактический параметр команды **\SkipSpace** – это «**A.**», а второй – «**B.**». Так как для второго параметра ограничителя нет, то он определяется по обычным правилам, то есть является токеном или группой. Группа, заключенная в фигурные скобки, никогда не разбивается на части. Поэтому внутри группы ограничитель параметра игнорируется. Например, макрокоманда

```
\SkipSpace {A. B.} Степанов
```

раскроется в **A. B.Степанов**, потому что первым параметром будет группа «**A. B.**» (фигурные скобки убираются после определения фактических параметров), а вторым – «**C**».

Вообще, правила раскрытия макропоследовательностей довольно просты, если в них не вмешиваются такие средства LaTeX-программирования, как команда **\expandafter**, условные операторы и т. п. После определения фактических параметров TeX просто подставляет их на место формальных (то есть на место символов **#1**...) в текст определения, после чего ставит полученный текст определения на место макрокоманды и параметров. После подстановки TeX продолжает «читать» документ с начала подставленного текста.

Продолжение следует.

Степанов Алексей Владимирович, доцент кафедры «Высшей математики №2» СПбГЭТУ («ЛЭТИ»).