Степанов Алексей Владимирович

ЯЗЫКИ РАЗМЕТКИ. ЧАСТЬ 2: ОСНОВНЫЕ СРЕДСТВА ФОРМАТИРОВАНИЯ

В этой статье мы разберем основные средства форматирования в языках LaTeX и HTML: переключение шрифтов, параметры абзацев, создание списков.

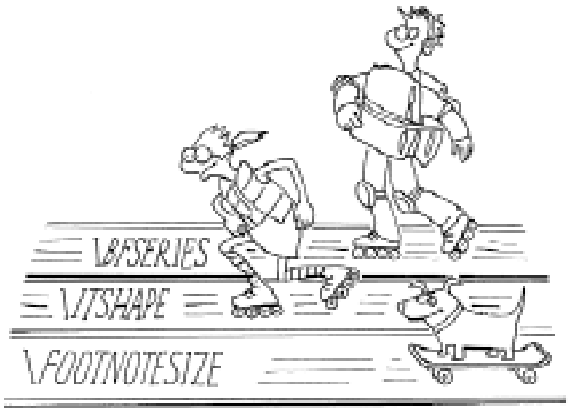
ПЕРЕКЛЮЧЕНИЕ ШРИФТОВ

В языке LaTeX определены следующие атрибуты шрифтов, которые можно менять независимо друг от друга.

1. **Размер.** Он переключается командами **\tiny**, **\scriptsize**, **\footnotesize**, **\small**, **\normalsize**, **\large**, **\Large**, **\LARGE**, **\huge** и **\Huge**.

2. **Семейство.** Определены 3 стандартных семейства шрифтов: обычный шрифт – **\rmfamily** (rm от слова Roman), шрифт без засечек – **\sffamily** (Sans Serif) и моноширный шрифт или шрифт пишущей машинки – **\ttfamily** (True Type).

3. **Насыщенность.** Стандартная насыщенность называется **\mdseries**, а полужирная – **\bfseries** (Bold Face).



4. **Начертание.** LaTeX предлагает 3 различных начертания: прямое – **\upshape**, курсивное – **\itshape** (Italic), наклонное – **\slshape** (Slanted) и капитель – **\scshape** (Small Capitals). *Капитель* – это начертание, при котором строчные буквы похожи на прописные буквы меньшего размера.

Приведенные команды переключения атрибутов шрифта действуют внутри группы (напомним, что группа в LaTeX ограничивается фигурными скобками, макросами **\begingroup...\endgroup** или любым окружением. Таким образом, если написать

```
{\bfseries полужирный {\itshape курсив}
\footnotesize маленький} обычный
```

то на печати получим что-то вроде

полужирный *курсив* маленький обычный

В языке LaTeX есть три варианта синтаксиса переключения шрифтов:

1. Командой:

```
команда_переключения_шрифта{текст}
```

2. Декларацией:

```
{набор_аттрибутов_шрифта текст}
```

3. Окружением:

```
\begin{свойство_шрифта} текст
\end{свойство_шрифта}
```

При этом команды переключения шрифтов выглядят следующим образом: **\textsf**, **\texttt**, **\textbf**, **\textit**, **\textsl** и **\textsc**. Как мы уже говорили, команды переключения шрифтов являются макрокомандами с одним аргументом. Команды переключения шрифтов удобнее для измене-

ния шрифта на небольшом участке текста, так как они автоматически вставляют курсивные поправки – небольшие дополнительные промежутки при переходе с курсива на прямой шрифт. Для отмены всех шрифтовых команд и перехода на основной шрифт документа используется декларация **\normalfont** или соответствующая команда **\textnormal**. Если шрифт нужно изменить на длинном участке текста, можно пользоваться соответствующим окружением. Для экономии памяти и времени трансляции лучше не пользоваться командами, аргументы которых занимают больше одного абзаца. Окружение, переключающее шрифт, также вставляет курсивную поправку. Декларативное переключение шрифтов удобно использовать при определении новых макрокоманд и окружений.

Как вы могли заметить, специального подчёркнутого шрифта в TeXe нет. Для подчёркивания используются специальные команды, причём линия подчеркивания может проходить по базовой линии текста или по нижнему краю букв (например, буква «р» выходит ниже базовой линии) или где вы захотите, но об этих командах мы поговорим позже. Вообще, подчеркнутый шрифт необычен для математических текстов, для печати которых изначально создавался TeX, поэтому подчеркивание не является стандартным средством.

На самом деле TeX выбирает реальный шрифт по заданным атрибутам на основе внутренних таблиц NFSS (New Font Selection Scheme). Эти таблицы нетрудно научиться модифицировать или даже создавать самому, прочитав, например, соответствующий раздел книги [2]. Если TeX по каким-то причинам не может выбрать шрифт с заданными атрибутами, например, из-за отсутствия шрифтового файла, указанного в таблице NFSS, то он старается подобрать шрифт, наиболее близкий к заданному. К сожалению, это один из немногих алгоритмов, которые не всегда выдают приемлемый результат. Так что любому TeXнарю приходится отслеживать наличие нужных шрифтов. Для единичного использования какого-нибудь экзотического шрифта вместо таблиц NFSS можно воспользоваться низкоуровневым интерфейсом нераскрываемой команды **\font**. Синтаксис этой команды таков:

```
\font\xxx=<имя_шрифтового_файла>
          [at <размер>] [scaled 1000k]
```

где **\xxx** – имя команды, **<размер>** – это величина, измеряемая в TeXовских единицах измерения, например, **cm** (сантиметры), **in** (дюймы) или **pt** (типографские пункты), а **k** – коэффициент масштабирования. Таким образом, написав в преамбуле

```
\font \eightitbf=wcmbi10 at 8pt
```

или

```
\font \eightitbf=wcmbi10 scaled 800
```

вы получите полужирный курсив, масштабированный с коэффициентом 0.8 из соответствующего шрифта гарнитуры wcm (Computer Modern Cyrillic Fonts в кодировке cp1251). После этого для использования этого определения достаточно написать

```
{\eightitbf это пример} использования
                  команды \verb|\font|
```

чтобы получить

это пример использования команды **\font**

Обратите внимание на фигурные скобки, окружающие группу, внутри которой действует определенная нами шрифтовая команда **\eightitbf**. Команда **\verb** означает дословное (verbatim) воспроизведение. Синтаксис этой команды:

```
\verb <символ><текст><символ>
```

где **<текст>** – это произвольная последовательность символов, которая может включать непечатные символы. При этом все они будут напечатаны без изменений моноширным шрифтом.

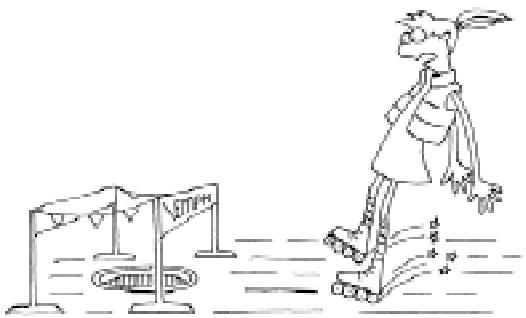
Сейчас мы рассматривали локальную разметку, то есть четкие указания, какой конкретный шрифт (точнее, шрифт с какими атрибутами) будет использован. Конечно, эти указания тоже можно переопределить с помощью таблиц NFSS или, написав что-нибудь типа

```
\renewcommand\textsl#1{\textit#1},
```

при желании использовать везде курсив вместо наклонного шрифта. Однако так делать не принято. Вместо того чтобы переопределять стандартные команды переключения шрифтов, принято использовать логическую разметку. Так как идея логической разметки проходит путеводной нитью через весь цикл статей, поясним сказанное подробнее.

Допустим, что вы не знаете, каким будет основной шрифт, окружающий некоторый фрагмент, но вам хочется его выделить. Эта ситуация встречается регулярно. Когда я пишу математическую статью, я еще не знаю, в каком журнале она будет опубликована, поэтому не знаю, будет ли формулировка теоремы напечатана прямым шрифтом или курсивом (стандарт AMS). В этом случае TeX предоставляет команду **\emph** (emphasize – выделить), а о том, как именно этот текст будет выделен, должен позаботиться стиль данного журнала. В стандартных стилях типа article команда **\emph** просто меняет начертание шрифта: если основной шрифт – курсив, то выделенный будет прямым, и наоборот. Хочется верить, что вы почувствовали разницу между локальной и логической разметкой. В приведенном примере логическая разметка указывает, что часть текста должна быть выделена, а как именно – определяется в стилевом файле. Если все авторы будут придерживаться логической разметки и не писать **\textbf** или **\textit** для выделения фрагмента текста, то все выделенные участки во всем журнале будут выглядеть одинаково!

Ситуация с переключением шрифтов в HTML очень похожа. Для задания атрибутов шрифта существуют следующие тэги локальной разметки:



- **<b>** – полужирный шрифт;
- **<i>** – курсив;
- **<u>** – подчеркнутый шрифт;
- **** с атрибутами:
  - **size**
  - **face**
  - **color**

Значениями атрибута **size** может быть либо абсолютная величина шрифта, определяемая числом от 1 до 7, либо относительная величина от –5 до +5 («+» писать обязательно). Атрибут **face** определяет гарнитуру шрифта и может принимать значения типа **"Times New Roman"**. Значения, содержащие пробелы, обязательно должны быть заключены в кавычки. Атрибут **color** определяет цвет текста и может принимать одно из предопределенных значений: **black**, **white**, **blue** и т. п. или задавать цвет при помощи шкалы RGB, например,

...,

что соответствует шестнадцатеричному коду цвета морской волны.

Кроме этого, существуют два тэга логической разметки:

- **** – выделить (emphasize) и
- **** (без перевода).

Оба, конечно, существуют для обозначения выделенного текста, но первый по умолчанию выделяет курсивом, а второй – полужирным шрифтом. Значения логической разметки могут быть переопределены в таблицах стилей. Таблицы стилей являются одним из средств программирования для языка HTML. Эта тема будет затронута позже.

РАЗМЕРЫ И ПРОМЕЖУТКИ

Для того чтобы рассказывать далее принципы локальной разметки, необходимо познакомиться со способами задания и единицами измерения длин. В TeX длины измеряются в типографских пунктах (**pt**), в сантиметрах (**cm**), миллиметрах (**mm**), дюймах (**in**) или в единицах измерения, зависящих от величины текущего шрифта: em или ex. Имеют место следующие равенства:

- **1cm = 10mm = 28.4526pt**
- **1in = 25.4mm = 72.27pt**
- **1em** = ширина буквы **m** текущего шрифта = **10pt**
- **1ex** = высота буквы **x** текущего шрифта = **4.30554pt**

При этом значения для **em** и **ex** приведены для шрифта по умолчанию (**cmr10**). Так как TeX является расширяемым языком, то можно определить свои единицы измерения с помощью команды **\newdimen**, например,

```
\newdimen\en  \en=0.5em
```

определит размер **\en** как половину ширины буквы **m** текущего шрифта. Однако это удобство не идет ни в какое сравнение с практикой использования растяжимых и сжимаемых промежутков. Такой промежуток можно определить с помощью команды

```
\newlength\xxx
```

и изменить его значение (по умолчанию оно равно **0pt ± 0pt**) одним из следующих способов:

- **\setlength\xxx{x plus y minus z}** или эквивалентный оператор
- **\xxx= x plus y minus z**, причем как «**plus y**», так и «**minus z**» может быть опущено.
- **\addtolength\xxx{x}**
- **\settowidth\xxx{text}**
- **\settoheight\xxx{text}**
- **\settodepth\xxx{text}**

Здесь **\xxx** – имя нового промежутка, **x**, **y**, **z** – размеры, а **text** – произвольный текст. Задание для **\xxx** значения **x plus y minus z** означает, что при форматировании текста TeX постарается сделать промежуток длиной **\xxx**, но при необходимости может растянуть его до **x+y** или сжать до **x-y**. Команда **\addtolength** добавляет размер **x** к промежутку **\xxx**, а остальные команды устанавливают **\xxx** равным ширине, высоте или глубине текста **text**. Для понимания последнего необходимо отметить, что любой кусок текста в TeX имеет привязку по вертикали, *базовую линию*. Высотой текста называется расстояние от базовой линии до верхнего края текста, а глубиной – расстояние от нижнего края до базовой линии. На самом деле, TeX вычисляет эти размеры, исходя из метрических характеристик шрифта, которые записаны в tfm-файле. TeX вообще не имеет дело с картинками, изображающими буквы. С ними работают программы, воспроизводящие dvi-файл на экране, при печати, или переводящие его в другой (скажем, pdf) формат.

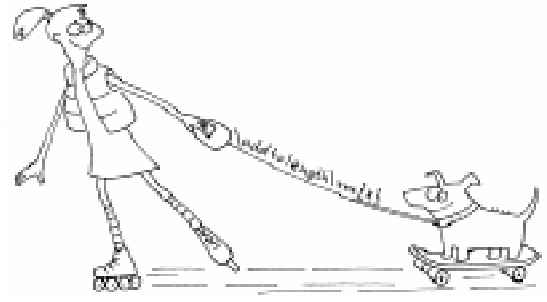
Необходимо знать следующие предопределенные в LaTeX промежутки:

- **\baselineskip**, **\lineskip** и **\lineskiplimit** (последний является размером, а не промежутком, то есть не допускает растяжения и сжатия), отвечающие за расстояние между соседними строками текста. Вычисляя межстрочный интервал, TeX использует следующий алгоритм: он старается сделать расстояние между базовыми линиями строк равным **\baselineskip**, однако, если при этом нижний край первой строки отступает от верхнего края нижней меньше, чем на **\lineskiplimit**, то**\lineskip** используется как промежуток между краями строк.
- **\smallskipamount**, **\medskipamount** и **\bigskipamount** – дополнительные вертикальные промежутки, вставляемые при помощи команд **\smallskip**, **\medskip** и **\bigskip**. По умолчанию, **\smallskipamount = 3 pt ± 1pt**, **\medskipamount = 6pt ± 2pt**, а **\bigskipamount = 12pt ± 4pt**.

В языке HTML размеры задаются в пикселях (по умолчанию) или в процентах от размера родительского элемента (например, окна браузера или таблицы).

РАЗМЕТКА АБЗАЦЕВ

Начало абзаца в TeX обозначается командой **\par** или оставленной пустой стро-



кой (два символа перевода каретки подряд, возможно разделенные пробельными символами). Абзацный отступ задается параметром **\parindent**, который, по умолчанию, равен **2em**. Для определения расстояния между абзацами используется параметр **\parskip**. По умолчанию, все абзацы выравниваются «по ширине». Для задания другого способа выравнивания можно использовать окружения **center**, **flushright** и **flushleft** или команды **\centering**, **\raggedleft** и **\raggedright**, соответственно. Для задания отступа от левого или правого края всех строк абзаца можно использовать параметр **\leftskip**, соответственно **\rightskip**. Кстати, команда **\raggedright** эквивалентна **\setlength\rightskip{0pt+1fil}**.

Команда **\parindent=0pt** отключает абзацный отступ. Однако иногда мы хотим напечатать только данный конкретный абзац без абзацного отступа (в стандарте AMS так принято набирать теоремы или определения). Для этого используется команда **\noindent**. Если вы хотите выделить абзац дополнительным вертикальным отступом, например перед теоремой, можете написать **\smallskip**, **\medskip**, **\bigskip** или даже **\vskip<промежуток>**. Если такая команда попала в начало страницы, то она игнорируется. При желании оставить вертикальный отступ от верхнего края (например, первой) страницы можно, если воспользоваться вариантом команды **\vskip** со звездочкой, то есть написать **\vskip*<промежуток>**, которая вставит вертикальный **<промежуток>**, независимо от того, куда попадет. В LaTeX многие команды имеют варианты со звездочкой, которые делают примерно то же самое, что и вариант без звездочки. С этим мы еще столкнемся, обсуждая команды секционирования и автоматическую нумерацию.

Иногда (при подготовке качественного печатного текста очень редко) хочется перейти на новую строку, не создавая новый абзац. Для этого предназначена команда **\newline** или просто ****. Причем последняя команда может иметь еще и необязательный аргумент – вертикальный отступ. Например, можно написать

```
\noindent
Это странный пример того,\\[-10pt] как
строчки налезают друг на друга.,
```

чтобы получить

> Это странный пример того
> как строчки налезают друг на друга.

Если же вы наоборот хотите, чтобы в каком-то месте текст не разрывался, то можете поставить символ **~** вместо пробела – этот символ является неразрывным пробелом. Если же вы хотите, чтобы часть материала обязательно появилась в одной строке, можете использовать конструкцию

```
\hbox{горизонтальный материал}
```

Все, что находится в горизонтальном боксе (**\hbox** – сокращение от horizontal box), будет напечатано на одной строке.

Алгоритм, по которому TeX разбивает абзац на строки, довольно сложен и основан на системе штрафов. За каждое нештатное действие типа переноса слова или увеличение межстрочного интервала ставится штраф, который программа старается минимизировать. Так как TeX формирует образ страницы, то штрафы по всем абзацам суммируются со штрафами за не самое удачное размещение вертикального материала (разрыв страницы близко к концу абзаца и т. п.) и минимизируется именно этот суммарный штраф. Существуют параметры TeX, позволяющие уменьшать или увеличивать штрафы за отдельные операции, однако изучение этих параметров далеко выходит за рамки настоящей статьи. Надо отметить, что TeX *никогда* не делает расстояние между словами больше положенного и *никогда* не разрывает слова или формулы в неположенном месте. Поэтому иногда при форматировании получаются переполненные строки (overfull hbox). Существует две возможности избежать переполненных строк: либо форматировать их вручную, добавляя промежутки в каких-то местах или разрывая строки, либо позволить программе разбираться с этим самостоятельно, изменив некоторые параметры разбив-

ки на строки. Я лично предпочитаю пользоваться вторым способом, используя следующие 4 параметра.

• **\spaceskip** – отвечает за расстояние между словами. Грубо говоря, написав **\setlength\spaceskip{.3em plus 1em}**, вы позволяете увеличивать межсловное расстояние до **1.3em**.

• **\tolerance** – принимает целые неотрицательные значения и определяет количество межсловных интервалов в абзаце, которые могут отличаться от оптимального. Изменяется командой **\tolerance=<число>**. По умолчанию равен 200. Разумные значения – от 50 до 9999. Как уже упоминалось, числа от 10000 TeX считает бесконечностью, таким образом, значение **\tolerance=10000** позволит делать сколь угодно жидкие строки.

• **\emergencystretch** – промежуток, который при необходимости может быть дополнительно вставлен в строку.

• **\hfuzz** – размер, на который разрешается делать строки длиннее.

Если я получаю переполненную строку в обычном абзаце, то обычно достаточно бывает написать:

```
{\tolerance=1000\emergencystretch=
1em\hfuzz=.5pt ...текст абзаца...},
```

и все переполнение исчезает. Хуже, когда в абзаце встречается слишком длинная формула, скажем, на полстроки. В этом случае мягкое изменение параметров может не помочь, а более жесткое привести к тому, что в какой-то строке останется три слова, и абзац будет выглядеть очень плохо. Что делать в этой ситуации, мы обсудим в разделе про набор математики.

В языке HTML абзац должен быть ограничен тэгом ** ... **. Этот тэг имеет единственный атрибут **align** со значениями **justify**, **left**, **right** или **center**. Все остальные параметры абзаца можно выставить только с помощью языка стилей, и это правильно в том смысле, что они не должны быть параметрами локальной разметки. В HTML существует неразрывный пробел ** **; и принудительный переход на новую строку – непарный тэг **
**.

Я так много сказал про форматирование абзацев в TeX, и так мало – в HTML, что это удивило даже меня самого. На самом деле, ничего удивительного нет: во-первых, к качеству текста в HTML не предъявляются столь же строгие требования, как в TeX – мало кто будет издавать печатную продукцию с помощью языка HTML; во-вторых, язык стилевых файлов TeX – это тот же самый TeX, и мы обсудили некоторые возможности, которые разумно использовать в стилевом файле, а язык стилевых файлов HTML называется CSS (Cascade Style Sheets Language) и будет обсуждаться в отдельном разделе.

СЧЕТЧИКИ TeX

Перед тем как обсуждать форматирование списков и таблиц, необходимо получить минимальные сведения о хранении и операциях с целыми числами в LaTeX. Целая переменная в TeX называется счетчик или count-регистр. Синтаксис счетчиков LaTeX немного отличается от базового синтаксиса TeX, при этом некоторые предопределенные счетчики TeX унаследованы в LaTeX, поэтому мы обсудим оба варианта. В TeX:

• **\newcount\n** – определелеление счетчика **\n**;

• **\advance\n by x** – увеличение **\n** на **x**, при этом **x** может быть счетчиком или числом или макропоследовательностью, раскрывающейся в число;

• **\multiply\n by x** – умножение **\n** на **x**;

• **\divide\n by x** – деление **\n** нацело на **x**;

• **\the\n** – вывод **\n** на печать арабскими цифрами.

В LaTeX:

• **\newcounter{n}[m]** – определяет **\newcount\c@n** (при этом, конечно, код категории символа **@** равен 11, иначе **\c@n** не являлось бы макропоследовательностью TeX), подчиненный уже существующему счетчику **m**; при этом **m** является необязательным параметром;

• для того чтобы синтаксис TeX стал доступен, смените код категории символа **@**, используя команды **\makeatletter** и **\makeatother**, то есть пишите **\makeatletter \multiply\c@n by 2\makeatother**, для того чтобы умножить LaTeX-овский счетчик **n** на 2;

• **\setcounter{n}{x}** устанавливает значение **n=x**;

• **\addtocounter{n}{x}**, **\stepcounter{n}** прибавляют к счетчику **n** число **x** (соответственно 1); также по команде **\stepcounter{n}** обнуляются все счетчики, подчиненные **n**;

• **\arabic{n}**, **\roman{n}**, **\Roman{n}**, **\alph{n}** и **\Alph{n}** выводят **n** на печать арабскими цифрами, строчными или прописными римскими цифрами, строчную или прописную букву латинского алфавита с номером **n**, соответственно.

В LaTeX предопределены несколько видов счетчиков, которые мы рассмотрим в соответствующих разделах. Сейчас упомянем только, что **page** – это счетчик страниц.

СПИСКИ

Для форматирования списков в LaTeX используются окружения **enumerate** и **itemize**. Синтаксис этих окружений почти одинаков:

```
\begin {itemize}
\item Первый элемент списка
\item Второй элемент списка
...
\end {itemize}
```

Маркер списка называется **\labelitemi** (для вложенных списков – **\labelitemii** и т. д.). По умолчанию, это – большая жирная точка. Чтобы переопределить это, нужно написать

```
\renewcommand\labelitemi{<новый_маркер>}
```

Если написать это определение внутри окружения, то оно будет действовать только для данного списка, если вне (например, в преамбуле или в стилевом файле) – то для всех списков **itemize** данного документа. Если хочется изменить маркер только данного элемента списка, можно поставить этот маркер в квадратных скобках после команды **\item**. Приведем пример вложенного списка.

```
\renewcommand\labelitemii{---} %%
                                длинное тире
\begin{itemize}
\item Первый пункт.
\item Второй пункт.
\item[] Не маркированный пункт.
\item[!!!] Нестандартно маркированный
                                      пункт.
\begin{itemize}
\item Первый подпункт.
\item Второй подпункт.
\item[2.3.] Нестандартно маркированный
                                   подпункт.
\end{itemize}
\end{itemize}
```

На выходе получим следующую картинку:

- Первый пункт.

  Не маркированный пункт

  – Первый подпункт.

  2.3. Нестандартно маркированный подпункт.

  – Следующий подпункт.

- Следующий пункт.

!!! Нестандартно маркированный пункт.

Без дополнительных определений возможно создание пяти уровней списков. Нумерованный список создается аналогично ненумерованному с помощью окружения enumerate. При этом при определении меток можно пользоваться счетчиками **enumi...enumv**, представлением счетчиков на печати **\theenumi...\theenumv** и мет-

ками **\labelenumi...\labelenumv**. Промежуточное звено этой иерархии введено потому, что оно используется также при формировании автоматических ссылок на пункты списка, которое будет рассмотрено в разделе про автоматическую нумерацию. Итак, давайте определим метки и напишем пример вложенного нумерованного списка.

```
\renewcommand\theenumi{\Roman{enumi}}
\renewcommand\theenumii{\alph{enumii}}
\renewcommand\labelenumii{\theenumi.
                          \theenumii.}
\begin{enumerate}
\item Первый пункт.
\item Второй пункт.
\item[] Не маркированный пункт.
\item[!!!] Нестандартно маркированный
                                  пункт.
\begin{enumerate}
\item Первый подпункт.
\item Второй подпункт.
\item[\arabic{enumi}.\arabic{enumii}.]
  Нестандартно маркированный подпункт.
\end{enumerate}
\end{enumerate}
```

Тогда на печати получим:

Надеюсь, что, глядя на номер 2.2., вы обратили внимание, что нестандартно нумерованные пункты и подпункты не увеличивают соответствующий счетчик. Любые отступы и выступы можно переопределить, добиваясь поразительных эффектов, однако это существенно выходит за рамки ознакомительной статьи. Все про создание и переопределение параметров списков можно прочитать в книге [1].

Теперь посмотрим, как создаются списки в HTML. HTML заметно «лаконичнее», чем TeX, по длине команд. Например, нумерованный список выглядит следующим образом:

```
<OL>
<LI>Первый</LI>
<LI>Второй</LI>
...
</OL>
```

Ненумерованный список отличается от нумерованного всего одной буквой: тэг называется **<UL>** вместо **<OL>**. Как и в случае форматирования абзацев, параметры списков фактически доступны только в языке CSS, поэтому мы отложим их рассмотрение на будущее.

I. Первый пункт.

II. Второй пункт.

Не маркированный пункт.

!!! Нестандартно маркированный пункт.

II.a. Первый подпункт.

II.b. Второй подпункт.

2.2. Нестандартно маркированный подпункт.

Продолжение следует

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