Семейство классов listok *

Илья Райко
rayko_i@179.ru

1 сентября 2020 г.

Аннотация

Классы, используемый нашей командой для вёрстки листков.

1 Введение

На данный момент у нас есть классы listok, written, ustn, test. listok используется для вёрстки листков, written и ustn используются для вёрстки письменных и устных собеседований, test используется для контрольных работ.

Разница между ними описана в следующей табличке

	listok	test	written	ustn
Заголовок	`\title{Title}`			
	Title	***Title. Вариант №num***		***… собеседование***
Задача	`\begin{problem}[Text]`			
	Задача num.problem*Text*	**Задача problem***Text*		
Пример	`\begin{example}[Text]`			
	Пример num.example*Text*	**Пример example***Text*		
Теорема	`\begin{theorem}[Text]`			
	Теорема num.theorem *(Text)*	**Теорема theorem** *(Text)*		

При этом у классов есть ещё некоторые индивидуальные особенности, которые будут описаны ниже.

2 Параметры классов

2.1 Формат бумаги

```
⟨common⟩\RequirePackage{kvoptions}
⟨common⟩\newcommand{\@paperfile}{}
⟨common⟩\DeclareOption{a4paper}{\renewcommand{\@paperfile}{a4.clo}}
```

С a4paper всё просто — напечатает одну копию на листе A4, ориентированном вертикально. Ширина текста будет 179mm, а высота 267mm. Текст выровнен по центру (вертикально и горизонтально).

```
⟨*a4⟩
    \RequirePackage[a4paper, width = 179mm, height = 267mm, footskip = 0mm, centering, includehe
    \PassOptionsToClass{a4paper,oneside}{article}
⟨/a4⟩
⟨common⟩\DeclareOption{a5paper}{\renewcommand{\@paperfile}{a5.clo}}
```

Чуть сложнее с a5paper. Он нужен для того, чтобы напечать два вертикальных A5 на одном горизонтальном A4. Признаться честно, я уже забыл, что делают конкретные команды, но это, кажется, работает.

```
⟨*a5⟩
    \setlength\paperheight {210mm}
    \setlength\paperwidth  {148mm}
    \PassOptionsToClass{a5paper,oneside}{article}
    \RequirePackage{pgfpages}
```

*Этот документ относится к listok v2.1, от 2019/01/09

```
    \RequirePackage{atbegshi}
    \setlength\hoffset{.4in}
    \setlength\oddsidemargin{-1in}
    \setlength\textwidth{130mm}
    \setlength\voffset{-25mm}
    \setlength\textheight{185mm}
    \pgfpagesuselayout{2 on 1}[a4paper,landscape,border shrink=5mm]
    \AtBeginShipout{%
        \pgfpagesshipoutlogicalpage{1}\copy\AtBeginShipoutBox%
        \pgfpagesshipoutlogicalpage{2}\box\AtBeginShipoutBox%
        \pgfshipoutphysicalpage%
    }
⟨/a5⟩
```

2.2 Дата и номер листка

`date` задаёт дату листочка в углу. Сейчас значение по умолчанию — сегодня. Бейте меня палками, если до сих пор не взлетает.

```
⟨*common⟩
\DeclareStringOption[\DTMToday]{date}
```

`num` — номер листочка. Влияет на номер задач.

```
\DeclareStringOption[1]{num}
```

Все остальные опции передаются классу `article`.

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\ProcessKeyvalOptions*
\LoadClass{article}
\input{\@paperfile}
\RequirePackage{iftex}
```

3 Движки

Вот тут начинается цирк с движками. Проблема заключается в том, что традиционный Web2C движок использует кодировки отличные от utf8 (OT2, T2A, T2B, T2C, etc.) Чем это плохо? В OT2 был очень странный порядок сортировки кириллических букв (Э, Ю, Ж, Й, Ё, Я, А, Б, Ц, Д, Е, Ф и т.д.), что выливается в отвратительные списки и предметные указатели.

Даже игнорируя это факт, мы получаем две проблемы: старый Web2C движок генерирует странный слой текст в pdf (зюковки и кракозябры вместо русского текста), можно использовать только шрифты поддерживающие кодировки T2. Проблемы в том, что был только один такой шрифт, в котором русские буквы имеют равную высоту. Этот шрифт распространялся пакетом `pscyr`, который не обновляется с конца 2000-х годов. И тут нас постигает совсем проблема: `pscyr` сегодня поддерживается только дистрибутивом MiKTEX.

Поэтому предлагается использовать расширенный движок LuaTEX. Если не вдаваться в подробности, то имеет место простая формула:

$$\text{LuaTEX} = \text{pdfTEX} + \text{Поддержка Unicode и OpenType шрифтов} + \\ + \text{e-TEX} + \text{Omega} + \text{встроенный интерпретатор lua}$$

Корректная работа классов семейства `listok` гарантируется при использовании LuaLATEX. В теории можно использовать и другие движки, но на свой страх и риск (на момент 19.01.2019 я не мог скомпилировать русский текст с pdfLATEX).

Движок определяется автоматически по имени запускаемой программы.

```
\ifLuaTeX
    \input{lua.clo}
⟨/common⟩
⟨*lua⟩
    \RequirePackage[TU]{fontenc}
    \RequirePackage[lutf8]{luainputenc}
    \RequirePackage{fontspec}
```

```
    \setmainfont{CMU Serif}
⟨/lua⟩
⟨common⟩\else
⟨pdf⟩\RequirePackage[utf8]{inputenc}
⟨pdf⟩\RequirePackage{cmap}
⟨*common⟩
    \input{pdf.clo}
\fi
```

4 Фишки

Далее идут подключённые пакеты

```
\RequirePackage[russian]{babel}
\RequirePackage[inline]{enumitem}
\RequirePackage{textcomp, multicol}
\RequirePackage{mathtext}
\RequirePackage{indentfirst}
\RequirePackage{amsmath, amssymb, amsfonts, amsthm}
\RequirePackage{mathtools, mathabx}
\RequirePackage{epstopdf}
\RequirePackage{graphicx}
\RequirePackage{forloop}
\RequirePackage{datetime2}
\RequirePackage{microtype}
\relax
\tolerance 4000
```

`\listok@name` задаёт имя листка, затем используется командой `\maketitle` для заголовка.

```
⟨listok⟩\newcommand{\listok@name}{    №\listok@num}
⟨test⟩\newcommand{\listok@name}{    №\listok@num}
⟨written⟩\newcommand{\listok@name}{}
⟨ustn⟩\newcommand{\listok@name}{    ,   }
\renewcommand{\maketitle}{%
    \begin{center}%
    {\Large \textit{\textbf{\underline{%
⟨listok⟩\@title %
⟨test⟩\@title. \listok@name %                .      №N
⟨written⟩
⟨ustn⟩
    }}}}
    \end{center}
}

\DTMsetup{datesep=.}
\DTMsetstyle{ddmmyyyy}
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
\renewcommand{\@oddhead}{
    \vbox{\hbox to\textwidth{\listok@name\hfil \strut
    \listok@date
    }\hrule}
}
\renewcommand{\@oddfoot}{}
```

4.1 Трансокеанизация

Команда	Вывод
`\emptyset`	$\emptyset$
`\le`	$\leqslant$
`\ge`	$\geqslant$
`\epsilon`	ε
`\phi`	φ

\N	$\mathbb{N}$
\Z	$\mathbb{Z}$
\Q	$\mathbb{Q}$
\R	$\mathbb{R}$
\Cx	$\mathbb{C}$
\Re z	$\operatorname{Re} z$
\Im z	$\operatorname{Im} z$
\Zm{p}	$\mathbb{Z}_p$
\Prob{Event}	$\mathbb{P}\left[Event\right]$
\ProbV{ξ}{Event}	$\mathbb{P}_{\xi}\left[Event\right]$
\Expect{ξ}	$\mathbb{E}\xi$
\pt{Text}	Text°
Text\point	Text°
Text\hard	Text^*

```
\renewcommand{\emptyset}{\varnothing}
\renewcommand{\le}{\leqslant}
\renewcommand{\ge}{\geqslant}
\renewcommand{\epsilon}{\varepsilon}
\renewcommand{\phi}{\varphi}
\newcommand{\N}{\mathbb{N}}
\newcommand{\Z}{\mathbb{Z}}
\newcommand{\Q}{\mathbb{Q}}
\newcommand{\R}{\mathbb{R}}
\newcommand{\Cx}{\mathbb{C}}
\renewcommand{\Re}[1]{\operatorname{Re}{#1}}
\renewcommand{\Im}[1]{\operatorname{Im}{#1}}
\newcommand{\Zm}[1]{\mathbb{Z}_{#1}}
\renewcommand{\Prob}[1]{\mathbb{P}\left [ #1 \right ]}
\newcommand{\ProbV}[2]{\mathbb{P}_{#1}\left [ #2 \right ]}
\DeclareMathOperator{\Expect}{\mathbb{E}}
\newcommand{\wdt}{\widetilde}
\newcommand{\pt}[1]{$\text{#1}^\circ$}
\newcommand{\point}{${}^\circ$}
\newcommand{\hard}{${}^\ast$}
```

4.2 Переопределение кванторов

Проблема: команды `\exists` и `\forall` задают просто символ, а не оператор или что-то ещё. Из этого следует, что на печати надо самим заботиться о пробелах после кванторной связки. Поэтому команды переопределены и ставят `\;` после связки.

К тому же доопределён макрос `\existsone`.

Команды `\existssym` и `\forallsym` задают символы кванторов в старом смысле.

```
\let\existssym\exists
\renewcommand*{\exists}[1]{\existssym #1~\;~}
\newcommand*{\existsone}[1]{\existssym! #1~\;~}
\let\forallsym\forall
\renewcommand*{\forall}[1]{\forallsym #1~\;~}
```

4.3 Остальное

`\closure` В «The Comprehensive LaTeX Symbol List» Скотт Пакин, рассуждая о различиях `\bar` и `\closure`, рекомендует промежуточный вариант, который придумал Энрико Грегорио и опубликовал во втором издании своего «Appunti di programmazione in LaTeX e TeX», который дословно воспроизведён тут

```
\newcommand{\closure}[2][3]{%
{}\mkern#1mu\overline{\mkern-#1mu#2}}
```

`abstract` Abstract имеет смысл использовать под разные комментарии. Текст печатается по центру курсивом.

```
\renewenvironment{abstract}{\quotation\itshape\centering}{\endquotation}
```

theorem
theorem*
lemma
lemma*
proposition
proposition*
corollary
corollary*
definition
definition*
note
problem
problem*
example
example*

Для вёрстки теорем, лемм, утверждений, следствий, определений и замечаний определены окружения. Вариант со звёздочкой делает нумерованную теорему, etc.

```
\newtheorem{theorem}{    }
\newtheorem*{theorem*}{    }
\newtheorem{lemma}{   }
\newtheorem*{lemma*}{   }
\newtheorem{proposition}{      }
\newtheorem*{proposition*}{      }
\newtheorem{corollary}{     }
\newtheorem*{corollary*}{     }
```

Попробую сделать крутые указания к задаче. Более того окружение само знает, что оно — указание к последней задаче. Новый стиль для amsthm теорем приследует следующую цель. Хотим отрисовывать звёздочку около сложной задачи и кружочек около обязательной. Для этого будем использовать аргумент, который обычно используют для названия теорем!

Можно использовать и для примеров.

```
\newtheoremstyle{problem}{0pt}{0pt}{\normalfont}{}{\bfseries}{.}{ }{\thmname{#1}\thmnumber{ #2}\
\theoremstyle{problem}
\newtheorem{example}{   }
\newtheorem*{example*}{   }
\newtheorem{problem}{   }
\newtheorem*{problem*}{   }
\newtheorem*{definition}{      }
\renewcommand \theproblem {%
⟨listok⟩     \listok@num.\@arabic\c@problem
⟨common&!listok⟩\@arabic\c@problem
}
\renewcommand \theexample {%
⟨listok⟩     \listok@num.\@arabic\c@example
⟨common&!listok⟩\@arabic\c@example
}
\renewcommand \thetheorem {%
⟨listok⟩     \listok@num.\@arabic\c@theorem
⟨common&!listok⟩\@arabic\c@theorem
}
```

\problems
\theme

\problems есть переписка с \section* из класса article. \theme — аналог \subsection* из класса article. Рекомендуется использовать, если задачи явно сгруппированы по какой-то теме. Команды \problems и theme есть только в классе listok

```
⟨∗listok⟩
\newcommand \problems{\@startsection{section}{1}{\z@}{-3.5ex \@plus -1ex \@minus -.2ex}{2.3ex \
\newcommand \theme{\@startsection{subsection}{2}{\z@}{-3.25ex\@plus -1ex \@minus -.2ex}{1.5ex \

⟨/listok⟩
```

vartab

Окружение vartab{⟨*amount*⟩} необходимо для набора таблиц различного размера. amount задаёт количество столбцов.

```
\newenvironment{vartab}[1]
{
    \begin{tabular}{*{#1}{|c} |c| } \hline
}{
    \end{tabular}
}
```

\conduit

\conduit печатает кондуит. В случае устного собеседования — это табличка в пять строк и \c@problem столбцов (то есть по одному столбцу на задачу). (Три оценки за каждую задачу и подпись). На листочке и контрольной — табличка с критериями оценки.

```
⟨∗ustn⟩
\newcounter{colidx}
\newcommand \conduit {

\vspace*{\fill}
\begin{center}
\begin{Large}
```

```
\begin{vartab}{\c@problem}
\hline
\forloop{colidx}{1}{\not{\value{colidx} > \c@problem}}{\; \arabic{colidx} \; &} \; $\Sigma$ \;
\forloop{colidx}{1}{\not{\value{colidx} > \c@problem}}{&} \\ \hline
\forloop{colidx}{1}{\not{\value{colidx} > \c@problem}}{&} \\ \hline
\forloop{colidx}{1}{\not{\value{colidx} > \c@problem}}{&} \\ \hline
\forloop{colidx}{1}{\not{\value{colidx} > \c@problem}}{&} \\
\forloop{colidx}{1}{\not{\value{colidx} > \c@problem}}{&} \\
\forloop{colidx}{1}{\not{\value{colidx} > \c@problem}}{&} \\ \hline
\end{vartab}
\end{Large}
\end{center}
}
```
⟨/ustn⟩
⟨∗listok | test⟩
```
\newcommand \conduit [4]{
\begin{center}
\begin{tabular}{|c|c|c|c|}
\hline
\multicolumn{4}{|c|}{        } \\
\hline
<<5>> & <<4>> & <<3>> & <<2>> \\ \hline
#1     & #2     & #3     & #4     \\ \hline
\end{tabular}
\end{center}
}
```
⟨/listok | test⟩

`enumitem` позволяет делать свои списки. Поэтому есть следующие списки:

`enumerate` Каждый пункт на новой строке, делает так: 1. , 2. , 3. , …

`itemize` Каждый пункт на новой строке, делает так: • , • , • , …

`enumerate*`, `itemize*` Тоже, что и двое выше, inline версия (все пункты в одну строчку)

`probenum` Каждый пункт на новой строке, делает так: **а.** , **б.** , **в.** , …

`probparts` Inline версия `probenum`, но делает так: **(а)** , **(б)** , **(в)** , …

`multienum` Есть один аргумент. Разбивает `probenum` на переданное число столбцов.

⟨∗common⟩
```
\AddEnumerateCounter{\Asbuk}{\@Asbuk}{ }
\AddEnumerateCounter{\asbuk}{\@asbuk}{ }
\setlist[itemize]{nosep, nolistsep}
\newlist{probenum}{enumerate}{1}
\newlist{probparts}{enumerate*}{1}
\setlist[enumerate]{nosep, nolistsep}
\setlist[probenum]{nosep, nolistsep, label = \textbf{(\asbuk*)}}
\setlist[probparts]{nosep, nolistsep, label = \textbf{(\asbuk*)}}
\newenvironment{multienum}[1]
{
    \begin{probenum}
    \begin{multicols}{#1}
}{
    \end{multicols}
    \end{probenum}
}
\endinput
```
⟨/common⟩