

Министерство науки и высшего образования
Российской Федерации

Национальный исследовательский ядерный университет
«МИФИ»

А. В. Кузнецов

ОСНОВЫ L^AT_EX

Учебное пособие

Москва 2021

УДК 681.322

ББК 32.97

К 89

Кузнецов А.В. **Основы \LaTeX** . Учебное пособие. М.: НИЯУ МИФИ, 2021. 364 с.

Книга, посвященная верстке в издательской системе \LaTeX , ориентирована как на начинающих авторов, так и на тех, для кого \LaTeX — привычный инструмент. В ней описаны ресурсы стандартного \LaTeX , достаточные для оформления дипломов, диссертаций, рукописей научных статей и книг.

Так как \LaTeX базируется на концепции программной верстки, в основу изложения положена логика работы компиляторов, выполняющих различные операции. Рассмотрены основные приемы верстки текста и формул, создания рисунков и таблиц, формирование библиографии, оглавления и предметного указателя.

Книга содержит большой объем справочной информации, собранной в таблицах, и обширный указатель, обеспечивающий быстрый и эффективный поиск интересующего материала.

Издание предназначено для студентов и аспирантов физико-математических специальностей, а также авторов, самостоятельно верстающих тексты.

Рецензенты РФЯЦ ВНИИТФ (г. Снежинск):

д-р физ.-мат. наук, проф. П.А. Лобода,

канд. физ.-мат. наук Е.А. Говрас,

канд. физ.-мат. наук Д.А. Грязных.

ISBN 978-5-7262-2680-7

© Национальный исследовательский
ядерный университет «МИФИ», 2021

© А.В. Кузнецов, 2021

Предисловие

В настоящее время нет нужды доказывать преимущества издательской системы \LaTeX в оформлении статей и книг научно-технической и особенно физико-математической тематики. Она прочно заняла ведущие позиции во всех издательствах, специализирующихся на выпуске научной литературы и периодики, так как сводит к минимуму редакционную обработку рукописей и существенно сокращает сроки их опубликования. В ее основе лежит концепция программной верстки, заложенная Дональдом Е. Кнудом, создавшим в 1978 г. язык программирования и компилятор \TeX , предназначенные для верстки текста, насыщенного математическими формулами [1]. Создавая документ, автор пишет программу, команды которой указывают, что нужно сделать с той или иной частью текста. Выполняя данную программу, компилятор \TeX верстает документ.

В 1984 г. Лэсли Лэмпорт разработал на основе \TeX удобный и компактный макроязык \LaTeX , кардинально минимизировавший количество команд разметки, необходимых для форматирования рукописи. Он обогатил концепцию программной верстки понятием стиля, или класса документа. Размечая текст, автор использует достаточно небольшой набор команд, имеющих различные настройки. Настройки помещаются в стилевой файл, определяющий вид сверстанного документа. Изменение стиля верстки не требует изменения текста программы, за исключением команды, загружающей нужный стиль.

Введенные дополнения оказались очень эффективными. К началу 90-х годов ведущие издательства научной литературы разработали собственные стили, отвечающие их полиграфическим традициям, и ввели \LaTeX в издательский процесс. Появилось большое количество пакетов, т.е. дополнительного программного обеспечения, существенно расширивших возможности \LaTeX . С годами их число неуклонно растет. Так как \TeX и \LaTeX развиваются и распространяются усилиями сообщества свободного

программирования, любой пользователь имеет право разработать новый пакет, решающий какую-то задачу верстки, зарегистрировать его и присовокупить к дистрибутиву \LaTeX , насчитывающему в настоящее время более трех тысяч пакетов. Как правило, такие пакеты не конфликтуют с ядром \LaTeX , однако могут конфликтовать между собой. С этой точки зрения все программное обеспечение следует разделить на стандартный \LaTeX , состоящий из ядра и небольшого числа пакетов, поддерживаемых и развиваемых основной командой разработчиков, и нестандартные пакеты, разработанные и поддерживаемые многочисленными энтузиастами. В практике выпуска научной периодики издательства используют только стандартный \LaTeX , остальные ресурсы отсекаются на стадии подачи рукописи в редакцию, поэтому нестандартные пакеты можно использовать лишь на свой страх и риск в документах, не предназначенных для публикации. Начиная с 1993 г. стандартом является версия $\text{\LaTeX} 2\epsilon$, ставшая за прошедшие годы очень стабильной. Именно она описана в данной книге.¹

В литературе [2–10] описано большое число пакетов, используемых для оптимизации различных аспектов верстки. Однако в подавляющем числе случаев автору, работающему над рукописью, вполне достаточно средств стандартного \LaTeX . Ему не нужно погружаться в изучение его богатых ресурсов. Его интересует не процесс, а результат, достигаемый наиболее коротким путем. Данное руководство предназначено послужить подспорьем таким пользователям. Оно не претендует ни на тщательность, с которой создание документа описано в книге С. М. Львовского «Набор и

¹ Много лет ведется разработка $\text{\LaTeX} 3$, направленная на полную совместимость с форматом unicode, гибкую работу со шрифтами, интеграцию ряда пакетов непосредственно в ядро \LaTeX и расширение возможностей компилятора за счет использования скриптов языка Lua. Одна из основных задач проекта — расширение возможности основных команд за счет введения дополнительных настроек без изменения их интерфейса, чтобы при переходе к новому стандарту сохранить возможность использовать ранее созданные документы. Многие наработки «незаметно» для пользователей уже внесены в $\text{\LaTeX} 2\epsilon$.

верстка в пакете \LaTeX » [2], ни на полноту описания стандартного \LaTeX в книге И. Котельникова и П. Чеботаева « $\text{\LaTeX} 2_{\epsilon}$ по-русски» [3], ни на фундаментальность обзора основных и дополнительных средств \LaTeX в серии книг М. Гуссенса, Ф. Миттельбаха, С. Ратца и А. Самарина [4–6]. Опираясь на опыт преподавания курса \LaTeX студентам Московского инженерно-физического института и общение с коллегами, активно использующими \LaTeX в своей практике, автор стремился свести к разумному минимуму набор обсуждаемых средств, считая его вполне достаточным для работы над статьей, дипломом и даже книгой. Насколько это удалось, судить Вам, читатель.

В книге изложены основы стандартного \LaTeX , позволяющие создавать и редактировать документы. Изложение разбито на две части, преследующие разные цели.

В первой описаны основные принципы верстки текста и формул, обсуждена работа с графикой, приведены методы создания таблиц и списка литературы. Изложение строится не на описании рецептов, как выполнить ту или иную операцию, а на логике работы компилятора, поняв которую, пользователь сам сможет найти оптимальное решение. Освоение материала первой части достаточно для создания документов, не требующих вмешательства в алгоритмы автоматической верстки.

Вторая часть адресована авторам, желающим освоить \LaTeX более глубоко. В ней обсуждаются дополнительные возможности, имеющиеся в базовом \LaTeX и предоставляемые стандартными пакетами, круг которых ограничен лишь теми, что используют в своей практике ведущие издательства технической литературы. Описанные ресурсы позволяют настраивать различные аспекты верстки, а также создавать большие документы и книги.

Автор выражает глубокую признательность А.А. Иванову, А.И. Маймистову и А.А. Синченко за полезные рекомендации и ценные замечания к рукописи.

А.В. Кузнецов

Часть I

Верстка в L^AT_EX

Глава 1

Создание документа

Документ \LaTeX представляет собой программу, выполняя которую, компилятор \TeX формирует документ по заданному шаблону. Чтобы избежать путаницы, формируемый при компиляции файл мы будем называть *документом*, а текст, редактируемый автором, — *программой*, или *рукописью*. Наиболее часто для представления документов используются форматы PostScript и PDF, обеспечивающие высокое полиграфическое качество и адекватность их воспроизведения в любой современной операционной системе. Рукопись можно редактировать в обычном текстовом редакторе, но намного удобнее пользоваться специализированными программами, которые имеются в любой современной операционной системе.

Для работы пользователю необходимы компиляторы, набор программных средств — пакетов в терминологии \LaTeX , и графическая оболочка, обеспечивающая комфортный набор текста программы вместе с управлением компиляторами и средствами просмотра готовых документов. Вдобавок к этому для формирования списков литературы к создаваемым документам рекомендуется иметь средства работы с библиографическими базами.

В настоящее время лучшей графической оболочкой является программа \TeXstudio , активно развивающаяся и реализованная

во всех операционных системах. В операционной системе Linux средства компиляции содержит дистрибутив `TeXLive`, а работу с PostScript- и pdf-документами обеспечивают многочисленные программы, входящие в состав оболочек KDE, Gnome и т. д. В операционной системе Windows для работы используется дистрибутив `MikTeX` и программы `Ghostscript`, `GSview`. Удобный интерфейс работы с библиографическими базами предоставляет программа `JabRef`, написанная на языке `java` и реализованная во всех операционных системах. Все перечисленные программы распространяются и используются свободно.

1.1. Установка и конфигурирование программ

`MikTeX` можно установить непосредственно из интернета в полном или базовом варианте. Полный установленный дистрибутив занимает несколько гигабайтов, при этом для работы нужна лишь малая его часть. Базовый комплект содержит только основные программные средства, а остальные добавляются по мере надобности. По умолчанию, недостающие компоненты загружаются из интернета, что не всегда удобно, так как работоспособность компиляторов зависит от доступности сети. Мы опишем процедуру установки как стандартного варианта, так варианта, обеспечивающего автономную работу `MikTeX`.

Стандартным путем является загрузка с сайта <http://MikTeX.org/> и запуск программы `basic-MikTeX-#-x?? .exe`, устанавливающей базовый комплект `MikTeX`. Здесь `#` обозначает текущую версию `MikTeX`, а `??` — разрядность машины пользователя. После установки вы получите готовый к работе базовый комплект `MikTeX`, который по мере необходимости будет автоматически устанавливать недостающие пакеты из интернета. Его следует подстроить, чтобы добавить шрифты, используемые для генера-

ции PostScript- и pdf-документов высокого качества.¹ Для этого запустите программу `MikTeXConsole`, откройте меню «`Packages`», в строке поиска введите название шрифтового пакета `cm-super`, после того как строка с его параметрами появится в основном окне программы, нажмите на нее правой клавишей мыши и в выпадающем меню выберите «`Install package`». После этого можно приступать к работе.

Установка автономного `MikTeX` предусматривает загрузку всех его компонентов, поэтому для нее потребуется надежное высокоскоростное интернет-соединение. Создайте новую папку с названием `MikTeX`. На сайте <http://MikTeX.org/> загрузите программу-установщик и сохраните ее в созданную папку. Запустите программу и выберите вариант установки «`Download MikTeX ⇒ Complete MikTeX`». Установщик загрузит все файлы дистрибутива, объем которого превышает два гигабайта. Закончив загрузку, вновь запустите программу, выберите меню «`Install MikTeX ⇒ Basic MikTeX`», и указав в качестве источника установки папку с компонентами `MikTeX`, установите базовый комплект. Окончив установку, запустите программу `MikTeXConsole`, выберите меню «`Settings`» и нажмите кнопку «`Change`» в конце строки `Packages are installed from`. В появившемся окне выберите «`Local package repository`» и нажмите кнопку «`Next`», после чего перейдите в папку с файлами `MikTeX`, выберите ее и нажмите кнопку «`Finish`». Выполнив указанные действия, получите готовую к работе базовую систему `MikTeX`, которая по мере необходимости будет автоматически устанавливать недостающие пакеты из папки, находящейся в вашем компьютере. Ее следует подстроить, как описано ранее.

Вдобавок к `MikTeX` нужно установить программы `Ghostsript`, `GSview`, `TeXstudio` и `JabRef`, закачав их текущие версии с сайтов:
<http://pages.cs.wisc.edu/~ghost/>

¹ Растровые (пиксельные) шрифты, используемые по умолчанию, имеют худшее качество, нежели поддерживаемые форматами PostScript и pdf векторные шрифты. Установка векторных шрифтов `cm-super` устраняет этот недостаток.

<https://texstudio.org>

www.jabref.org.

Программу TeXstudio лучше устанавливать последней, чтобы она автоматически настроила взаимодействие MikTeX и GSview. После ее установки загрузите русский словарь, используемый для проверки орфографии набираемого текста. Для этого запустите TeXstudio и в меню «Options ⇒ Configure TeXstudio» выберите «Language Checking», открыв ссылку «LibreOffice», найдите и скачайте последнюю версию русского словаря, а затем, нажав кнопку «ImportDicionary. . .», установите его. Если вы преимущественно работаете с русскими текстами, имеет смысл загружать этот словарь при запуске TeXstudio. Это можно сделать, выбрав его в меню «Default Language».

Установив перечисленные программы, вы получите полный набор средств для работы с ЛАТ_EX. Чтобы сделать работу комфортной, рекомендуем поближе познакомиться с интерфейсом графической оболочки и подстроить ее под свои потребности. Остановимся кратко на ресурсах, необходимых в первую очередь.

Рабочее окно программы TeXstudio показано на рис. 1.1. В его левой части находится многофункциональная панель, управляемая расположенными слева иконками. По умолчанию активирована навигационная панель (верхняя иконка), в которой показывается структура разделов документа и метки рисунков, таблиц, формул и т.д. Выбирая нужную метку или раздел, можно переключиться в соответствующую позицию программы. При этом раздел автоматически загружается, если программа разбита на несколько файлов. В следующей панели (вторая иконка) навигация осуществляется по закладкам, которые можно создать в тексте. Остальные панели предназначены для ввода различных конструкций и команд, хранящих множество разных символов. Чтобы облегчить свою дальнейшую работу, ознакомьтесь с их содержанием, активировав соответствующие иконки. Ввод команд и конструкций с помощью графических меню оградит вас от опечаток в именах команд и обусловленных ими ошибок компиляции.

The screenshot shows the TeXstudio interface with the following components:

- Structure Panel:** Lists document sections: test.tex, LABELS, BIBLIOGR..., Текст, Формулы, Графика, Таблицы (highlighted), and Литература.
- Editor Window:** Contains LaTeX source code for a table of contents and bibliography.


```

7 & 14 & 21 & 28 & 35 & 42 & \bf 49 & 56 & 63\\
8 & 16 & 24 & 32 & 40 & 48 & 56 & 64 & 72\\
9 & 18 & 27 & 36 & 45 & 54 & 63 & 72 & 81

\end{tabular}
\end{table}

\section{Литература}

Имеются средства форматирования таблиц, работы с гиперссылками и
многие другие
возможности. Описание всего этого вы найдете в книгах
\cite{Gussens99, Gussens01, Gussens02, Lvovsky03, Kotelnikov04,
Gratzer-book, Baldin-book, Belyakov-book, Rozhenko-book}.

\bibliography{LaTeXBooks}

\tableofcontents

\end{document}

```
- Messages Panel:** Shows the compilation process:


```

Process started: pdflatex.exe -synctex=1 -interaction=nonstopmode "test".tex
Process exited normally

```
- Preview Window:** Displays the rendered document content:
 - Text: "зацы и слова. Слова зацы пустыми строками. В словами и абзацами можество пробелов или пустых на форматирование. Разит во время компиляции этом TeX сам определяет между словами и абзаца-
 - Image: A tiger's head illustration.
 - Text: "Рис. 1"
 - Section: "IV. Формулы"
 - Text: "ения могут находиться мер, это $|x| = \sqrt{x^2}$. Сложив отдельный абзац."
 - Text: "Имеются средства фоляющие, например, вспо"
 - Text: "если $i \neq j$, при $i = j$."
 - Text: "дномеровать,"
 - Text: "Таблица I. Жирным шристых чисел."
 - Table:

	\times	2	3	4
$+$	2	4	6	8
$\cos^2 x,$	3	6	9	12
 - Equation: (1)

Рис. 1.1. Рабочее окно программы TeXstudio

Меню «LaTeX» и «Math» содержат основные средства верстки текста и формул. В меню «Wizards» можно быстро создать основу нового документа, сделать заготовку таблицы и вставить иллюстрацию.

В меню «Options ⇒ Configure TeXstudio» находится окно настроек графической оболочки. Чтобы получить доступ ко всем параметрам настройки, активируйте флажок «Show Advanced Options» в левом нижнем углу данного окна. Обширный набор настроек очень хорошо структурирован, поэтому можно легко найти интересующую настройку, если она предусмотрена. Меню «General» позволяет установить шрифт интерфейса TeXstudio, а в меню «Editor» можно выбрать шрифт окна редактора.

По умолчанию, при наборе открывающей скобки редактор автоматически добавляет к ней закрывающую, а если выделен фрагмент текста, заключает его в скобки. Сняв флажок «Auto Complete Parenthesis» в меню «Adv. Editor» окна настроек, можно изменить режим ввода на традиционное редактирование, при котором скобки вводятся как обычные символы.

В нижней строке окна редактора находятся два меню, позволяющие настраивать кодировку редактируемого файла и выбирать словарь, используемый для проверки орфографии текста. По умолчанию, TeXstudio настроен на работу с файлами в кодировке UTF-8, которую без особой нужды лучше не менять.

Если после загрузки текст превращается в абракадабру, установите правильную кодировку файла и повторите загрузку. Для этого в меню кодировок в нижней строке окна редактора или меню «Edit ⇒ Setup Encoding» выберите нужную кодировку² и нажмите клавишу «Reload With». После того как текст станет выглядеть нормально, снова откройте меню, выберите кодировку UTF-8 и нажмите клавишу «Change To», тогда файл будет сохраняться в кодировке UTF-8. Если хотите сохранить файл в другой кодировке, сделайте то же для нее.

² Обычно проблема возникает при загрузке файлов в кодировке windows-cp1251, поэтому прежде всего следует опробовать ее.

Если скомпилированный документ содержит абракадабру, проверьте кодировку текста, объявленную в программе (см. разд. 2.7). Она должна совпадать с кодировкой файла.

1.2. Компиляция и просмотр документа

Организуя процесс компиляции, `TeXstudio` записывает программу в файл и указывает компилятору путь к нему. Таким образом, внесенные в текст изменения автоматически сохраняются при компиляции.

В подготовке документа участвует несколько программ, каждая из которых делает свою часть работы. При работе они генерируют ряд файлов и посредством них обмениваются информацией. Для успешного выполнения своей функции каждая из них нуждается в данных, подготовленных другими программами. Чтобы сверстать документ в окончательном виде, зачастую нужно запустить несколько программ в строго определенном порядке.

Блок-схема процесса компиляции представлена на рис. 1.2. Генерируемые файлы наследуют имя компилируемого документа, а их расширение определяется содержащейся в них информацией. Серым цветом на блок-схеме выделены файлы, загружаемые как внешние ресурсы, которые компиляторы не изменяют.

Основным компилятором является программа `pdfTeX`. Для ее запуска используется программа `pdfLaTeX`, передающая компилятору требуемые настройки. При компиляции в качестве внешних ресурсов помимо текста программы и рисунков загружаются следующие файлы: форматный файл `LaTeX` (`•.fmt`), класс документа (`•.cls`) и пакеты (`•.sty`) и (`•.tex`). В скобках указаны расширения имен файлов, которые будут обозначаться таким же образом и далее. Кроме этого загружаются файлы, созданные в ходе предшествующих компиляций: оглавление (`•.toc`), предметный указатель (`•.ind`), библиография (`•.bbl`) и информация, необходимая для нумерования разделов, формул, цитируемых источников и т. д. (`•.aux`).

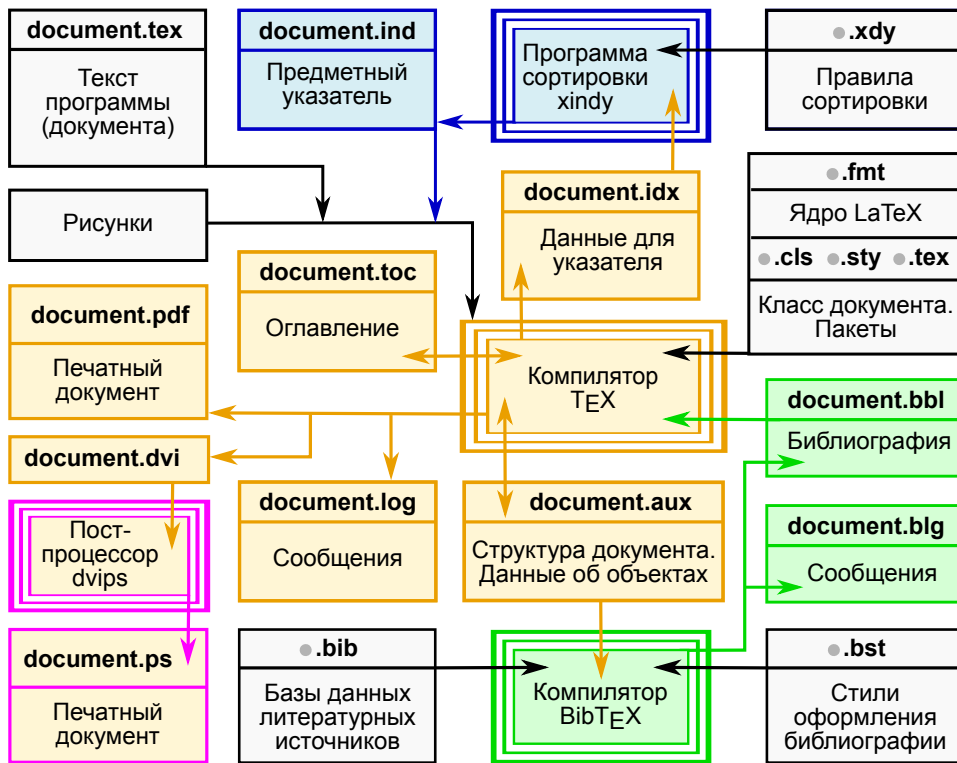


Рис. 1.2. Блок-схема процесса компиляции

В процессе работы компилятор TeX создает документ (`•.pdf` или `•.dvi`) и генерирует ряд служебных файлов, содержащих сведения для предметного указателя (`•.idx`), оглавление (`•.toc`) и отчет о компиляции (`•.log`). В файл с расширением `•.aux` заносится информация о порядке следования рисунков, таблиц, цитируемых источников и других объектов в тексте программы.

Если документ сохраняется в формате `dvi`, программа `dvips` конвертирует его в PostScript.

Для создания библиографии используется программа `BibTeX`. Из файла с расширением `•.aux` она считывает список цитируемых источников, загружает библиографические базы (`•.bib`) и стиль оформления библиографии (`•.bst`), формирует список литературы и сохраняет его в файл с расширением `•.bbl`. Отчет о компиляции направляется в файл с расширением `•.blg`.

Предметный указатель составляет программа `MakeIndex` или `xindy`. Из файла с расширением `•.idx` она считывает список терминов, формирует указатель и сохраняет его в файле с расширением `•.ind`, записав отчет о компиляции в файл с расширением `•.blg`.

Таким образом, помимо самого документа компиляторы генерируют значительное число дополнительных файлов, поэтому для каждого нового документа рекомендуется создать собственную рабочую папку. Файлы, генерируемые в процессе компиляции, наследуют имя файла с текстом рукописи, как показано на рис. 1.2 на примере файлов с именем «document».

Файлы с расширениями `•.aux`, `•.toc`, `•.bbl` и `•.ind` используются в последующих компиляциях. При смене класса документа или изменении набора загружаемых пакетов они являются потенциальным источником ошибок, так как содержащиеся в них команды могут стать неопределенными, или начнут конфликтовать с другими командами. В таком случае все ранее сгенерированные файлы³ следует удалить и начать компиляцию «с чистого листа». В TeXstudio для этого предусмотрено меню «Tools ⇒ Clean Auxiliary Files». Меню «Tools» содержит также средства запуска

³ Файлы с расширениями указанными на рис. 1.2 в иконках с фоном отличным от серого.

различных комбинаций компиляции, описанные в прил. А. На панели задач имеются иконки запуска компилятора \LaTeX и цепочки компиляций, формирующей все компоненты документа, включая библиографию и предметный указатель.

\TeXstudio имеет окно просмотра pdf-документов, расположенное справа от окна редактора (см. рис. 1.1). После компиляции в нем показывается страница, содержащая редактируемый абзац,⁴ который недолго «подсвечивается».

Окна редактирования и просмотра между собой связаны. Наведя курсор и нажав клавишу «Control» вместе с левой клавишей мыши, можно переключиться в соответствующее место текста программы или документа. Тот же переход можно совершить, нажав правую клавишу мыши и выбрав «Go to Source» или «Go to PDF» в появляющемся меню.

В ОС Windows для просмотра документов не следует пользоваться программой **AdobeReader**. Открывая файл, она блокирует его изменение, что вызывает сбой работы компилятора \TeX , который не может сохранить сверстаный документ.

1.3. Основные понятия и синтаксис \LaTeX

Программа \LaTeX состоит из текста, команд и комментариев.⁵

Текст разбивается на слова, разделяемые пробелами, и абзацы, разделяемые пустыми строками. Между словами можно поставить любое количество пробелов, а между абзацами любое количество пустых строк, при компиляции они игнорируются. Символы перевода строки считаются обычными пробелами.

Компилятор сам разбивает текст на строки и устанавливает промежутки между словами и строками, исходя из заданного стиля верстки. Дополнительные пробелы следует использовать для наглядного представления структуры программы и конструкций

⁴ Абзац, в котором находится курсор окна редактора.

⁵ Строго говоря, имеются еще и формулы, которые также состоят из символов (текста), команд и комментариев и потому, с точки зрения синтаксиса, не являются особыми объектами.

Л^AT_EX. Сравните, например, два варианта кода, один из которых не очень удобен для восприятия:

```
$$\delta_{ij}=\begin{cases}1,&i=j\\0,&i\neq j\end{cases}$$,
```

а второй — прост и понятен:

```
$$ \delta_{ij} = \begin{cases} 1, & i=j, \\ 0, & i\neq j. \end{cases} $$$
```

$$\delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases}$$

Оба они после компиляции дают одинаковый результат.

Здесь и далее конструкция

`часть программы` \rightsquigarrow `часть документа`

используется, чтобы показать, как будет выглядеть часть документа после компиляции `части программы`.

Комментарий открывается символом процента % и ограничивается концом строки. В комментариях, состоящих из нескольких строк, каждая из них должна начинаться символом процента. Текст и команды, находящиеся в комментарии, при компиляции полностью игнорируются. Комментарии могут находиться в любом месте программы, в том числе в аргументах и параметрах команд. Комментарий не является пробелом и потому не разрывает текст, но попытка разбить комментарием имя команды вызовет ошибку. Проиллюстрируем сказанное следующим примером:

```
Текст %      А  
будет не% комментарий       $\rightsquigarrow$       Текст будет непрерывен!  
прерывен!% пропадет!
```

1.3.1. Команды и окружения

Признаком команды является обратный слэш, за которым следует ее имя, параметры и аргументы:

```
\samplecommand[параметр]{аргумент}.
```

Имя команды состоит из латинских букв и звездочки *, которая может стоять в конце него. Аргументы команд заключаются в

фигурные скобки, а параметры в квадратные. Прописные и строчные буквы в именах различаются, поэтому, например, команды $\backslash AA \rightsquigarrow \mathring{A}$ и $\backslash aa \rightsquigarrow \mathring{a}$ производят разные символы.

Л^AT_EX не имеет какого-либо специального признака окончания имени команды. Им служит любой символ, отличный от латинских букв и звездочки. После имени с равным успехом могут стоять как пробел,⁶ так и скобки, знаки препинания, цифры, русские буквы и т. д., в любом случае компилятор распознает команду.

Наличие и количество аргументов и параметров вводится определением команды. Ее строение не ограничено каким-либо шаблоном. Л^AT_EX не регламентирует порядок следования аргументов и параметров, однако есть общие правила, которым они подчиняются. Во-первых, аргументы и параметры всегда следуют за именем команды. Во-вторых, они могут содержать другие команды, пробелы, комментарии и т. д. И наконец, между именем, аргументами и параметрами допускаются пробелы и комментарии, но не должно быть пустых строк.

Аргумент содержит информацию, без которой команда не может быть выполнена, или объекты, с которыми она должна совершить какое-то действие. Если аргумент не выделен явно фигурными скобками, вместо него используется символ (или команда), следующий непосредственно за командой, например, $\backslash \text{textbf}\AA \rightsquigarrow \mathring{A}$. Если полученный таким образом аргумент не удовлетворяет требованиям команды, компилятор выдаст сообщение об ошибке. В частности, команда $\backslash \text{hspace } 1\text{cm}$ некорректна. Ее аргументом служит длина, характеризующаяся величиной и размерностью, а число 1, получаемое в качестве аргумента, размерности не имеет и потому длиной не является.




Описанный алгоритм действий компилятора называется *правилом наследования аргумента*.

⁶ Следует учитывать, что компилятор автоматически убирает пробел, следующий за именем команды, поэтому символ, заданный командой и отделенный от текста одним пробелом, сольется с текстом. Чтобы избежать этого, после такой команды лучше поставить пару фигурных скобок {}.

Параметр служит для настройки каких-либо аспектов выполнения команды. Если параметр не используется, его можно опустить вместе с квадратными скобками. В таком случае используется значение параметра, заданное по умолчанию при определении команды.

В качестве примера использования параметров и аргументов приведем команду:

`\rule[смещение]{длина}{толщина}`,

рисующую линию заданной длины и толщины и смещенную при необходимости относительно базисной линии строки. Команда `\rule[5pt]{25pt}{1pt}` рисует смещенную вверх горизонтальную линию , команда `\rule[-5pt]{1pt}{10pt}` — смещенную вниз вертикальную линию , а команда `\rule{6pt}{6pt}` — несмещенный квадратик .

Служебные символы

Таблица 1.1

•	Назначение	Команда
<code>\</code>	признак команды	<code>\textbackslash</code>
<code>#</code>	параметр команды	<code>\#</code>
<code>{</code>	открывающая скобка	<code>\{</code>
<code>}</code>	закрывающая скобка	<code>\}</code>
<code>_</code>	нижний индекс	<code>_</code>
<code>^</code>	верхний индекс	<code>\textasciicircum</code>
<code>\$</code>	математическая мода	<code>\\$</code>
<code>&</code>	разделение ячеек	<code>\&</code>
<code>%</code>	комментарий	<code>\%</code>
<code>~</code>	неразрывный пробел	<code>\textasciitilde</code>

Для часто выполняемых действий используются односимвольные команды, или служебные символы, упрощенный синтаксис которых является исключением из правил. Они собраны в первой колонке табл. 1.1, помеченной символом •, во второй описано их назначение, а в последней приведены команды, используемые, чтобы напечатать сами символы. Часть из них также имеют нестандартный синтаксис.

Отметим, что команды нижнего и верхнего индекса имеют аргумент, подчиняющийся приведенному выше правилу наследования. Сравните, $\$x^2_{ij}\$ \rightsquigarrow x^2_{ij}$ и $\$x^2_{ij}\$ \rightsquigarrow x^2_{ij}$.

Одной из основных конструкций ЛАТ_EX является окружение:

```
\begin{имя окружения} [параметры] {аргументы}
    область действия окружения
\end{имя окружения}
```

Оно формируется командами `\begin` и `\end`, охватывающими часть программы, являющейся областью действия окружения. Их аргумент — имя окружения, определяющее операцию, производимую окружением. Для имен справедливы требования, предъявляемые к именам команд.

Открывающая команда `\begin` может иметь дополнительные аргументы и параметры, а у закрывающей команды `\end` они отсутствуют, так как ее задача лишь ограничить область действия производимой операции. Окружения позволяют более наглядно представить структуру текста программы и в этом их основное назначение.

Некоторые команды имеют окружения-аналоги, например команда `\itshape курсив` \rightsquigarrow *курсив* и окружение:

```
\begin{itshape}
    Для верстки большого          Для верстки большого объ-
    объема текста лучше           ема текста лучше пользо-
    пользоваться окружениями.    ваться окружениями.
\end{itshape}
```

Команды используются для обработки небольших фрагментов текста в пределах строки или абзаца, тогда как окружения более удобны при верстке нескольких абзацев текста.

Области действия команд тесно связаны с группами — одним из базовых понятий ЛАТ_EX. Прежде чем приступить к верстке, компилятор считывает множество определений команд и значений переменных, задающих их параметры. Лишь малая часть

переменных является глобальными и еще меньшая — статическими, т. е. сохраняющими неизменными свои значения во время компиляции. Подавляющая же часть представляет собой локальные динамические переменные, значение которых определено только внутри группы.

Группой является аргумент и параметр команды, окружение, математическая формула, ячейка таблицы или просто часть текста, заключенная в фигурные скобки. С точки зрения компилятора, сам текст программы также представляет собой группу, так как заключен в окружение `document`. Группы вкладываются друг в друга: текст программы содержит окружения и команды, в аргументах и параметрах которых находятся другие окружения и команды со своими аргументами и параметрами, и т. д. Уровень вложенности может быть очень велик. При выходе из вложенной группы измененные локальные переменные восстанавливают свои значения, а вновь введенные переменные и команды становятся неопределенными.

Исходя из сказанного, можно условно выделить два типа команд. Область действия одних ограничена их собственным аргументом. Они создают группу (аргумент) и внутри нее меняют динамическую переменную, например параметр шрифта, как в этом случае: `\textbf{при}мер` \rightsquigarrow **пример**. К данному типу команд относятся и окружения, область действия которых также является группой.

Команды другого типа, называемые *декларациями*, действуют в пределах группы, в которой находятся, начиная с позиции, в которой они стоят. Декларации не действуют на предшествующий текст, но изменяя значение какой-либо динамической переменной, меняют режим верстки следующего за ними текста, вплоть до выхода из группы.

Поясним сказанное примером, в котором группы формируются фигурными скобками, а декларации `\it`, `\sf`, `\large` и `\small` изменяют динамические переменные, управляющие начертанием и размером шрифта. Выделим скобки и команды фоном, а текст приведем в уже сверстанном виде:

{ Для начала создадим группу, вложенную в обычный текст. `\it \large` *Здесь изменим шрифт на курсив большого размера.* `{\sf\small` Создадим вложенную группу и установим рубленый шрифт малого размера. `}` *Выйдем из вложенной группы. Восстановился большой курсив.* `}` А теперь, выйдя из первой группы, возвратимся к исходному шрифту.

Некоторые декларации изменяют значение глобальных переменных, область действия которых не ограничивается группой. Вносимые ими изменения действуют до конца компиляции, или до новой установки значения переменной с помощью декларации. Такие декларации будем называть *глобальными*.

1.4. Структура документа

1.4.1. Преамбула. Класс документа и пакеты

Текст программы начинается с преамбулы, в которой задается класс документа, перечисляются загружаемые пакеты, настраиваются существующие команды и определяются новые. Завершает преамбулу команда `\begin{document}`.

Команда

```
\documentclass[ параметры]{класс документа}
```

загружает класс документа и осуществляет настройку основных параметров верстки. Класс можно задать только один, а параметры перечисляются списком через запятую. Например, для статьи, набираемой в две колонки шрифтом 12pt, можно указать

```
\documentclass[a4paper, 12pt, twocolumn]{article}
```

Научные издательства разработали большое число собственных классов, соответствующих их стилю и полиграфическим традициям. В основном они базируются на стандартных классах \LaTeX : разработчики адаптируют существующие конструкции под свои нужды и при необходимости добавляют небольшое число новых. Синтаксис основных конструкций и команд \LaTeX , как

правило, не меняется при замене одного класса документа на другой, поэтому изучение классов и программных средств стандартного L^AT_EX позволяет в дальнейшем самостоятельно освоить и использовать любой новый класс документа. Перечислим основные стандартные классы⁷ и параметры их настройки.

СТАНДАРТНЫЕ КЛАССЫ L^AT_EX

article — наиболее часто используемый класс, предназначенный для верстки научных статей.

report — класс, предназначенный для верстки отчетов.

proc — класс, предназначенный для верстки трудов конференций. Основан на классе **article**, но отличается тем, что текст печатается в две колонки.

book — класс, предназначенный для верстки книг. Характеризуется наиболее полным списком разделов. Обычно имеет оглавление, указатель, колонтитулы и двусторонний вывод с разными ширинами полей четных и нечетных страниц.

Следующие параметры позволяют настроить основные параметры верстки непосредственно при загрузке стандартных классов.

ПАРАМЕТРЫ НАСТРОЙКИ СТАНДАРТНЫХ КЛАССОВ

10pt | **11pt** | **12pt** — размер шрифта документа. По умолчанию используется **10pt**.

letterpaper | **legalpaper** | **exclusivepaper** | **a4paper** | **a5paper** | **b5paper** — размер страницы. По умолчанию используется **letterpaper**. Параметры **a5paper** | **b5paper** не поддерживаются классом **proc**.

⁷ Для верстки писем и презентаций имеются классы **letter** и **slides**, практически не используемые в настоящее время.

landscape — альбомная ориентация страницы. По умолчанию используется портретная ориентация.

draft | **final** — маркировка неполных или переполненных строк. В режиме **final**, используемом по умолчанию, проблемные строки не маркируются.

oneside | **twoside** — размеры полей четных и нечетных страниц. По умолчанию для всех классов, кроме **book**, используется **oneside** — поля одинаковы. Параметр **twoside** не используется в классе **slides**.

openright | **openany** — стиль разбиения на главы, которые могут начинаться только с правой (**openright**) или с любой (**openany**) страницы. По умолчанию первый метод использует класс **book**, а второй — **report**.

onecolumn | **twocolumn** — вывод текста в одну и две колонки. По умолчанию для всех классов, кроме **proc**, используется одна колонка.

titlepage | **notitlepage** — вывод заголовка и аннотации. По умолчанию в классах **proc** и **book** заголовок и аннотация выносятся на отдельную страницу, что соответствует **titlepage**, а в классе **article** они располагаются в начале документа. Параметр **titlepage** не поддерживается классом **proc**.

fleqn — выравнивание формул по левому краю. По умолчанию они центрируются.

leqno — при использовании данного параметра номера формул располагаются слева. По умолчанию он следует за формулой.

openbib — верстка списка литературы в «открытом стиле», в котором логические блоки цитируемого источника, разделяемые командами `\newblock`, начинаются с новой строки.

Возможности ЛАТ_EX существенно расширяются библиотеками-пакетами, которые выполняют дополнительные настройки команд и конструкций, позволяют оперировать графическими объектами и гипертекстовыми ссылками, вводят команды для реализации новых действий. В настоящее время насчитывается более тысячи пакетов и число их неуклонно растет, так как каждый пользователь вправе написать свой пакет, сделать к нему описание, зарегистрировать и выложить в СТАН-архив.⁸ Описание многих пакетов содержат книги [2–6], а подробная документация ко всем пакетам, установленным в MikTeX, находится в папке:

```
c:\Program Files\MikTeX #.#\doc\latex
```

Пакеты нужно разделить на стандартные и нестандартные. Первые реализуют действия, необходимые для всех классов документов, они рекомендованы к общему употреблению, так как не конфликтуют между собой и со стандартными классами.

Нестандартными пакетами можно пользоваться только на свой страх и риск, и хотя некоторые из них упрощают верстку, их не стоит включать в свой арсенал. Если пакет не рекомендован в документации класса, скорее всего его загрузка будет запрещена при компиляции документа в издательстве, что может вызвать прекращение компиляции.

Загружает и настраивает пакет команда

```
\usepackage [параметры] {имя пакета}
```

В ее аргументе можно указать список пакетов, перечисленных через запятую. Загружать списком следует лишь пакеты, не имеющие настроек.

Классы и пакеты могут самостоятельно (неявно) загружать и конфигурировать другие необходимые для их работы пакеты. Сообщение о загрузке каждого пакета выдается на консоль и записывается в log-файл.

⁸ СТАН (Comprehensive TeX Archive Network) — сеть общедоступной информации по Т_EX'у можно найти в интернете по адресу: www.ctan.org

Приведем пример простейшего документа. Слева расположен текст программы, а справа — сверстанный документ:⁹

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[russian]{babel}
\begin{document}
Текст документа.
\end{document}
```

Место для заметок и вариантов.

Верстаемый текст располагается после преамбулы в окружении `document`. Обратите внимание, что фраза «Место для заметок и вариантов.» отсутствует в сверстанном документе. Текст, находящийся за командой `\end{document}`, не обрабатывается. В находящемся за ней «остатке» файла можно хранить заготовки или варианты текста, чтобы использовать их в дальнейшем. Если компиляция дает большое число ошибок, с которыми трудно разобраться, ограничьте компилируемую часть программы, переместив команду `\end{document}` ближе к ее началу. Это облегчит работу над ошибками.

Перечислим пакеты, обеспечивающие поддержку русского языка, графики и расширенные ресурсы верстки формул:

<code>\documentclass[12pt]{article}</code>	размер шрифта
<code>\usepackage[a4paper,%</code>	размер страницы
<code>text={180mm, 260mm},%</code>	ширина и высота текста
<code>left=15mm, top=15mm]{geometry}</code>	размеры пустых полей
<code>\usepackage[utf8]{inputenc}</code>	кодировка текста
<code>\usepackage{cmap}</code>	кодировка pdf-документа
<code>\usepackage[russian]{babel}</code>	поддержка русского языка
<code>\usepackage{textcomp}</code>	текстовые символы
<code>\usepackage{indentfirst}</code>	корректировка отступов
<code>\usepackage{amssymb}</code>	математические символы
<code>\usepackage{amsmath}</code>	математические конструкции
<code>\usepackage{graphicx}</code>	поддержка графики

⁹ Далее будем использовать символ \rightsquigarrow только в примерах, находящихся в строке, и опускать его в остальных случаях.

Рассматриваемые далее возможности верстки предполагают загрузку всех пакетов этого минимальный набора, назначение которых обсуждается по мере необходимости.

1.4.2. Параметры страницы

Удобный и простой способ управления параметрами страницы дает пакет `geometry`. Число его настроек очень велико. Перечислим лишь те, что используются наиболее часто.

`landscape` | `portrait` — ориентация страниц.

`a4paper` | `a5paper`. . . — стандартный размер страниц. Определены практически все известные стандарты.

`paperwidth=...`, `paperheight=...` | `papersize={w,h}` — страницы произвольного размера. При использовании `papersize` в фигурных скобках указывается ширина и высота страниц.

`textwidth=...`, `textheight=...` | `text={w,h}` — ширина и высота текста.

`left=...`, `right=...` | `hmargin={l,r}` | `hcentering` — ширина левого и правого поля. При использовании `hcentering` поля имеют одинаковую ширину.

`top=...`, `bottom=...` | `vmargin={t,b}` | `vcentering` — высота верхнего и нижнего поля. При использовании `vcentering` поля имеют одинаковую высоту.

`centering` — эквивалент одновременного использования `hcentering` и `vcentering`.

`total={ширина, высота}` — полная ширина и высота всех компонент страницы.

`includefoot`, `includehead`, `includeheadfoot` — включение верхнего (`includehead`) и нижнего (`includefoot`) или обоих (`includeheadfoot`) колонтитулов в размеры `total`.

`onecolumn` | `twocolumn` — вывод текста в одну и две колонки.

`columnsep=...` — ширина поля между колонками.

1.4.3. Титульная часть

В начале программы обычно располагаются команды, содержащие заголовок, список авторов, дату создания и аннотацию документа. За ними следует команда `\maketitle`, обеспечивающая вывод этих сведений в верстаемый документ:

```
\begin{document}
\title{Заголовок}
\author{Автор, или перечень авторов}
\date{Дата создания документа}
\begin{abstract}
  Аннотация ...
\end{abstract}
\maketitle
```

Открывает документ заголовок, помещаемый в аргумент команды

```
\title{заголовок}.
```

Длинный заголовок можно поделить на строки командами `\\`.

Список авторов вместе с перечислением названий организаций и их адресов указывается в аргументе команды

```
\author{список авторов}.
```

Длинный список можно поделить на строки командами `\\`. Можно также использовать несколько команд `\author`, разбив список на группы.

В большинстве классов после списка авторов автоматически выводится текущая дата. При необходимости ее можно настроить командой

```
\date{дата}.
```

Текущую дату генерирует комбинация команд `\date \today`, используемая, если автоматический вывод не предусмотрен. Произвольную дату можно ввести вручную, указав ее в аргументе команды. Использование пустого аргумента подавляет автоматический вывод даты.

Команды `\title`, `\author` и `\date` игнорируются, если в программе отсутствует команда `\maketitle`, формирующая титульную часть документа. Оформление титульной части отличается от остального документа, например при двухколоночном наборе она занимает ширину страницы, а не колонки, а при использовании параметра `titlepage` стандартного класса выносится на отдельную страницу. Команда `\maketitle` указывает позицию, отделяющую титульную часть от остального текста. Она должна стоять после команд `\title`, `\author` и `\date`.

Окружение `abstract` формирует аннотацию. Ее положение зависит от класса документа. Во многих классах (например, `revtex4-2` Американского физического общества [11] и `elsarticle` Издательской корпорации Elsevier [12]) она входит в титульную часть и потому должна находиться перед `\maketitle`. Стандартный класс `article` создаст титульную страницу с аннотацией, если она находится перед `\maketitle`; и поместит ее после титульной части в начале первой колонки текста, если окружение `abstract` стоит после команды `\maketitle`.

1.4.4. Разделы

Документ ЛАТ_ЭX разбивается на ряд смысловых частей, которые обособляются в виде разделов документа. Разделы разбиваются на семь уровней, каждому из которых соответствует своя команда разбивки. Число доступных уровней разделов зависит от класса документа.

Уровни разделов стандартных классов и команды разбивки представлены в табл. 1.2. Ее первая и последняя колонки содержат команды, а кружки отмечают возможность их использования в документах данного класса. Как видно из таблицы, наиболее широким набором разделов обладают классы `book` и `report`. В книгах основным уровнем являются главы, которые могут разбиваться на разделы уровня `\section` и более мелкие части, или объединяться в части `\part`. Статья разбивается на разделы уровня `\section` и ниже.

Команды разбивки

Команда	book	report	article	proc	Команда
<code>\part[●]{●}</code>	○	○			<code>\part*{●}</code>
<code>\chapter[●]{●}</code>	○	○			<code>\chapter*{●}</code>
<code>\section[●]{●}</code>	○	○	○	○	<code>\section*{●}</code>
<code>\subsection[●]{●}</code>	○	○	○	○	<code>\subsection*{●}</code>
<code>\subsubsection[●]{●}</code>	○	○	○	○	<code>\subsubsection*{●}</code>
<code>\paragraph[●]{●}</code>	○	○	○	○	<code>\paragraph*{●}</code>
<code>\subparagraph[●]{●}</code>	○	○	○	○	<code>\subparagraph*{●}</code>
<code>\appendix</code>	○	○	○	○	

Команды разбивки перечислены в таблице в порядке вложенности уровней. Разделы верхних уровней могут содержать любое количество разделов более низких уровней, при этом с каждым уровнем ассоциирован свой счетчик. При создании раздела к счетчику его уровня добавляется единица, а счетчики более низких уровней сбрасываются, и нумерация всех вложенных разделов начинается заново. Таким образом, \LaTeX автоматически нумерует главы, разделы, параграфы и т. д. Команда `\label{имя метки}`, используемая непосредственно после команды разбивки, ассоциирует имя метки с номером раздела.

Чтобы создать раздел, достаточно поставить в его начале команду разбивки и пометить ее:

```
\section[0 друзьях]{Слоны мои друзья!}\label{s:слоны}.
```

Первая команда генерирует номер и выводит его вместе с заголовком, а вторая обеспечивает возможность ссылки на него.

Команды разбивки имеют аргумент и параметр, например:

```
\chapter[краткий заголовок]{заголовок}.
```

В аргумент помещается заголовок раздела. Слишком длинный заголовок можно поделить на строки командами `\\`. В параметре

задается краткое название раздела, используемое в оглавлении и колонтитулах. При отсутствии параметра краткое название заменяется заголовком.

Команды разбивки имеют дополнительный вариант с именем, завершающимся символом *. Команды «со звездочкой» предназначены лишь для вывода заголовка. Они не имеют параметра и не создают новый раздел. Номер перед ним не ставится, значения счетчиков не меняются, а сам заголовок не попадает ни в оглавление, ни в колонтитулы.

Декларация `\appendix` отделяет приложения от основной части документа. Она сбрасывает все счетчики, начиная с уровня `\chapter` в классах `book` и `report`, или `\section` в классах `article` и `proc`. Кроме этого несколько изменяется действие самих команд `\chapter` или `\section`. Во-первых, для вывода значений их счетчиков используются заглавные латинские буквы. Во-вторых, при создании приложения сначала выводится заголовок со стандартным названием «Приложение» (Appendix) и номером, а потом — название раздела без номера.

Проиллюстрируем сказанное примером:

```
\appendix
\section{Название}\label{•}
  Текст приложения\dots
\subsection{Заголовок}\label{•}
  Продолжение\dots
```

Приложение А

Название

Текст приложения...

A.1 Заголовок

Продолжение...

1.4.5. Оглавление

Команда `\tableofcontents` инициирует создание оглавления. При ее наличии компилятор, собрав названия разделов и вычислив номера страниц, на которых они созданы, формирует оглавление и сохраняет его в файле с расширением `•.toc`. При последующих компиляциях команда `\tableofcontents` замещается стандартным заголовком «Оглавление» (Table of Contents)

и считанным из `toc`-файла оглавлением, а сам файл обновляется. Таким образом, изменение структуры разделов в программе отразится в оглавлении документа лишь на следующем проходе компиляции.

1.4.6. Метки, ссылки, счетчики

Л^AT_EX имеет универсальную команду-метку:

```
\label{имя метки},
```

которой можно пометить объект, чтобы потом сослаться на него. Аргументом является `имя метки`, а объектом может быть формула, раздел, рисунок, таблица и просто позиция в тексте, в которой находится эта команда. Метка автоматически связывается с текущим значением счетчика страниц `и`, в зависимости от контекста, счетчика формул, рисунков и т. д. Информация о соответствии меток и счетчиков заносится в `aux`-файл. Сослаться на метку можно, используя команды:

```
\ref{имя метки} и \pageref{имя метки}.
```

Первая из них выводит номер помеченного объекта, а вторая — номер страницы, на которой он находится. Имеется также команда:

```
\eqref{имя метки},
```

предназначенная для вывода номера формулы, который она автоматически заключает в круглые скобки.

Ранее упоминалось, что для связывания меток и ссылок используются данные, хранящиеся в `aux`-файлах. Поэтому для «разрешения» ссылок необходимы, как минимум, две компиляции: при первой список меток и значений счетчиков сохраняется в файл, а при второй команды ссылок извлекают эту информацию. Если метка размещена в «плавающем» объекте (см. разд. 3), для корректного разрешения ссылок требуется три прохода компиляции: на первом в `aux`-файл записывается значение счетчика, соответствующее порядку следования объекта в программе; на

втором оно корректируется с учетом его положения в сверстанном документе и снова сохраняется в файл; лишь на третьем номер, выводимый в документ, становится правильным.

При загрузке пакета `hyperref` все ссылки преобразуются в гиперссылки.

Рисунки, таблицы, формулы и другие объекты автоматически подсчитываются во время компиляции. Каждый объект имеет собственный счетчик, обычно называемый так же, как и сам объект.

Список основных счетчиков приведен в табл. 1.3. В первой колонке перечислены счетчики страниц, рисунков, таблиц и уравнений, а также счетчик связанных уравнений `parentequation`, имеющих общий номер `equation`. Во второй и третьей колонке представлены счетчики разделов и сносок в основном тексте, `footnote`, и на министраницах (см. разд. 2.5), `mpfootnote`. В четвертой колонке находятся счетчики нумерованных списков (см. разд. 5.1 и 9.7.2) с первого до четвертого уровня.

Список основных счетчиков

Таблица 1.3

Прочие		Разделы		Списки
<code>figure</code>	<code>footnote</code>	<code>part</code>	<code>subsubsection</code>	<code>enumi</code>
<code>table</code>	<code>mpfootnote</code>	<code>chapter</code>	<code>paragraph</code>	<code>enumii</code>
<code>equation</code>	<code>parentequation</code>	<code>section</code>	<code>subparagraph</code>	<code>enumiii</code>
<code>page</code>		<code>subsection</code>		<code>enumiv</code>

Установить счетчик в заданное значение позволяет команда:

```
\setcounter{счетчик}{значение} .
```

К примеру, так можно сбросить счетчик примечаний:

```
\setcounter{footnote}{0}.
```

Используя команду:

```
\addtocounter{счетчик}{приращение} ,
```

можно сдвинуть счетчик на нужную величину. Отрицательное приращение уменьшает его значение, поэтому с помощью ко-

манды `\addtocounter{page}{-10}` можно вернуть нумерацию на десять страниц назад.

Текущее значение счетчика выводит команда

```
\thesчетчик .
```

Для независимых счетчиков выводятся только их значения, например `\thepage` \rightsquigarrow 34. К значениям подчиненных счетчиков обычно впереди добавляются значения внешних. Например, номер данного раздела `\thesubsection` \rightsquigarrow 1.4.6, а значения внешних счетчиков равны `\thesection` \rightsquigarrow 1.4 и `\thechapter` \rightsquigarrow 1. Подробнее о подчиненности счетчиков см. разд. 6.4.

Некоторые команды, связанные с обработкой счетчиков и ссылок на них, являются *хрупкими*. В тексте они обрабатываются корректно, но, находясь в аргументах или параметрах других команд и окружений,¹⁰ могут вызвать ошибки компиляции.

Ситуация с некорректной обработкой хрупких команд может возникать, если они оказываются в *перемещаемом* аргументе, который записывается в один из служебных файлов и используется при последующих компиляциях (см. рис. 1.2). Коллизии с хрупкими командами можно ожидать в названиях разделов, в подписях рисунков и таблиц, в аргументах команд, находящихся в плавающих объектах.

По мере совершенствования кода L^AT_EX список хрупких команд сокращается, и потому он не приводится, но некоторые из них обсуждаются далее.

Команда

```
\protect ,
```

изменяя алгоритм действий компилятора, позволяет «защитить» хрупкие команды и устранить ошибки. Она не имеет аргументов и ставится непосредственно перед «защищаемой» командой. Команду `\protect` можно использовать и с обычными командами, кроме тех, что хранят длины, или изменяют значения счетчиков. Это не принесет вред, однако может избавить от ошибок, если

¹⁰ В случае окружений это справедливо также и для области их действия.

они не связаны с несовместимостью команд, порой возникающей при работе с нестандартными пакетами.

1.4.7. Примечания

Примечания к тексту оформляются в виде сносок, выводимых внизу страницы.¹¹ Текст примечания помещается в аргумент команды

```
\footnote [номер] {текст} .
```

Сноски подсчитываются и нумеруются. По умолчанию новой сноске присваивается текущее значение счетчика `footnote`, но параметр команды `\footnote` позволяет указать любой другой номер. В стандартных классах `book` и `report` сноски нумеруются по главам, а в остальных используется сквозная нумерация. Примечания министраниц (см. разд. 2.5) нумеруются отдельно с помощью собственного счетчика `mpfootnote`.

Команда `\footnote` генерирует в тексте ссылку на сноску и выводит саму сноску на страницу, но выполнить обе эти операции вместе можно не всегда. При создании некоторых объектов связь с текущей страницей отсутствует, поэтому сноску вывести некуда. Например, в таблице `tabular` или плавающем объекте (см. гл. 3) команда `\footnote` генерирует лишь номер сноски, а сама она теряется. В таких случаях примечание можно сделать в два этапа с помощью команд:

```
\footnotemark [номер] ,  
\footnotetext [номер] {текст} .
```

Команда `\footnotemark` используется внутри объекта, чтобы сгенерировать номер сноски, а команда `\footnotetext` ставится снаружи непосредственно после объекта, чтобы вывести ее текст.


Команда `\footnotemark` является хрупкой, поэтому при появлении проблем ее следует защитить командой `\protect`.

¹¹ Сноски отделяются от основного текста чертой и печатаются мелким шрифтом.

Глава 2

Верстка текста

Верстая документ, компилятор решает задачу оптимального распределения объектов на площади страницы. Объекты представляют собой прямоугольники, в терминологии Т_ЕX называемые *боксами*. Боксы привязываются к базисным горизонтальным линиям и разделяются пробелами. Таким образом формируются строки.

Боксы могут иметь различную внутреннюю структуру. Простейший из них представляет собой букву или слово. Боксами являются также и загружаемая из файла иллюстрация , таб-

лица

а	с
е	п

 или строчная формула $\int x dx = x^2/2 + C$. Структура бокса может быть очень сложна, но будучи сформирован, он встраивается в текстовую строку как простой прямоугольник. Это означает, что обсуждаемые далее приемы верстки применимы не только непосредственно к тексту, но и для управления положением иллюстраций, таблиц и т. д.

Текст разбивается на абзацы. Верстая абзац, компилятор нарежет строки одинаковой длины. Заполняя строку, он «нанизывает» боксы на базисную линию до тех пор, пока их длина вместе с пробелами не превысит длину строки. Если крайний бокс можно разделить, часть его будет перенесена на следующую строку. Если

бокс представляет собой единое целое, компилятор оценит возможность опустить его в следующую строку, увеличив пробелы в текущей.

Сверстанный абзац встраивается в текущую страницу и, если она переполняется, часть строк переносится на новую страницу. Сначала на ней окажется мало строк, поэтому компилятор попытается перенести часть текста с предыдущей страницы, увеличив интервал между абзацами, оценит возможность перераспределить текст за счет изменения позиций плавающих объектов (см. разд. 3.3 и 9.4) и т. д.

В процессе компиляции оптимизируется не только результат выполнения отдельных операций, но и верстки в целом. Добавление одной строки может изменить распределение текста на нескольких страницах, поэтому пока документ не набран полностью, не стоит обращать внимание на погрешности автоматической верстки. Исправлять их следует в самом конце, окончив редактирование текста. Именно эта работа является главной при подготовке публикации.

2.1. Длины

Боксы, строки, страницы и другие объекты имеют размеры, измеряемые в «длинах». Между буквами, словами, строками, абзацами и т. д. имеются пустые промежутки, или пробелы, каждый из которых также имеет какую-то длину. Длина \LaTeX является очень гибкой величиной. Длины могут быть абсолютными и относительными, фиксированными и упругими.

Простейшей является фиксированная длина, задаваемая значением и размерностью. Основные размерности длин приведены в табл. 2.1. Значение представляет собой действительное число, в котором можно опустить нулевую целую или дробную часть, например 1cm , $+1.5\text{pt}$ или $-.2\text{mm}$. При этом единичное значение обязательно указывается, так как сама размерность длиной не является.

Таблица 2.1

Основные единицы длин

•	Значение	•	Значение
mm	миллиметр (2.845 pt)	in	дюйм (25.40 mm)
cm	сантиметр (0.39598 in)	pt	пункт (0.351 mm)
em	ширина буквы «М» текущего шрифта	ex	высота буквы «х» текущего шрифта

Таблица 2.2

Основные длины

Параметры страницы		Абзацы	Прочие
<code>\paperwidth</code>	<code>\textwidth</code>	<code>\parindent</code>	<code>\baselineskip</code>
<code>\paperheight</code>	<code>\textheight</code>	<code>\parskip</code>	<code>\columnsep</code>
<code>\voffset</code>	<code>\hoffset</code>	<code>\linewidth</code>	<code>\fill</code>

Упругие длины имеют сжимаемость и растяжимость, указываемые следующим образом: `12pt plus 4pt minus 4pt`. Такая длина может изменяться от `8pt` до `16 pt`. Использование упругих длин в пробелах, разделяющих боксы и абзацы, позволяет оптимизировать заполненность строк и страниц.

Относительные длины задаются в единицах `em` и `ex`, привязанных к текущему размеру шрифта, или кеглю, как принято называть его в полиграфии. Величина `1em` соответствует ширине заглавной буквы «М», а `1ex` — высоте прописной буквы «х». Абсолютная величина относительной длины вычисляется в соответствии с текущим кеглем при ее использовании. В конструкциях \LaTeX относительные длины позволяют согласованно регулировать размер шрифта, пробелов и межстрочного интервала, который задается длиной, меняющейся с изменением кегля.

Чаще всего длины хранятся в командах. Во время компиляции определен ряд длин разного масштаба. Часть из них, используемая наиболее часто, приведена в табл. 2.2. Первые две колонки содержат ширину и высоту страницы и текста, а также длины, позволяющие регулировать высоту верхнего (`\voffset`) и ширину левого (`\hoffset`) пустого поля на краю станицы.

В третьей колонке находятся ширина отступа в начале абзаца `\parindent`, величина дополнительного интервала между абзацами `\parskip` и межстрочный интервал `\baselineskip`. В последнюю колонку собраны ширина строки `\linewidth`, которая при верстке в две колонки отличается от ширины текста, ширина зазора между колонками `\columnsep` и длина `\fill`, обладающая нулевой естественной величиной и бесконечной растяжимостью. Примеры использования длин приводятся далее.

Изменить длину, хранящуюся в команде, позволяют декларации:

```
\setlength{длина}{значение} ,  
\addtolength{длина}{добавка} .
```

Первая устанавливает новое значение длины, а вторая изменяет ее на величину добавки.

Команда `\thedлина` печатает значение длины, например `\the\linewidth` \rightsquigarrow 321.51613pt.

2.2. Верстка абзацев

Рассмотрим алгоритм верстки абзацев и некоторые методы изменения формата строк.

Уже говорилось, что строки формируются из боксов, разделяемых упругими пробелами. Каждый бокс имеет высоту, ширину и глубину, как показано на рис. 2.1. При встраивании в строку бокс опускается относительно базисной линии на свою глубину, а его высота задает размер над базисной линией.

Простейший бокс — символ, например буква. Набранное слово также становится боксом, высота и глубина которого соответствуют максимальной высоте и глубине составляющих его букв. Буквы «склеиваются» упругим «клеем», поэтому длина бокса-слова может отличаться от суммарной ширины букв.¹

¹ В словах некоторые буквы сближаются, чтобы промежуток между ними не выглядел слишком рыхлым. Сравните `\tbox{у}лица` \rightsquigarrow улица и улица.

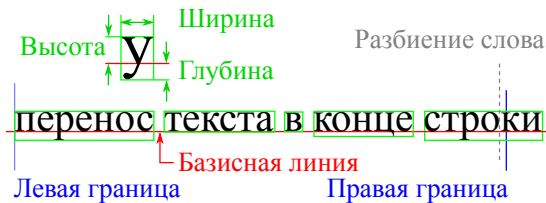


Рис. 2.1. Структура строки текста. Сплошными линиями обозначены границы боксов, а также базисная линия и границы строки. Вверху показаны параметры бокса с привязкой к базисной линии и место разбиения последнего бокса в соответствии с таблицей переноса

В абзаце, состоящем из текста, набранного шрифтом одного размера, межстрочный интервал одинаков. Однако расстояние между строками, содержащими боксы с большой высотой и глубиной, увеличивается в соответствии с размерами боксов, как это показано на с. 3б.

2.2.1. Выравнивание строк

Верстая абзац, Т_ЭХ выравнивает его левую и правую границу. Нарезая текст на строки одинаковой длины, он использует таблицу переносов и автоматически разбивает слова, оказавшиеся в конце строки.

Таблицы переносов предоставляет пакет **babel**. Язык указывается в параметре загрузающей его команды `\usepackage`. Таблица переносов используется для разбиения слов. Если перенос выбран неудачно, или слово отсутствует в таблице, перенос нужно настроить вручную, указав командами `\-`, как можно разбить слово (например, `ку\‑рѣ\‑ез`). Если слово окажется в конце строки, будет выбран оптимальный перенос, в остальных же случаях команды `\-` игнорируются. Продemonстрируем это следующим образом. Длинное «слово» *xx*, заведомо отсутствующее в таблице переносов, разобьем на группы из трех букв и получим вполне приемлемый результат:
`xxx\‑xxx…\‑xxx\‑xxx` \rightsquigarrow *xx-xxxxxx*.

Иногда, напротив, требуется подавить перенос, или сделать небольшой фрагмент текста «монокотным». Для этого его нужно поместить в бокс командой

```
\mbox{текст}.
```

В таком боксе можно разместить одно или несколько слов, формул и других объектов. Если длина бокса окажется велика, компилятор попытается перенести его в следующую строку, увеличив пробелы в текущей, но если она окажется очень «рыхлой», Т_EX, сжав пробелы до минимума, оставит бокс в текущей строке, которая из-за этого может вылезти за границу страницы.

В ряде случаев, чтобы слова не оказались в разных строках, их следует «склеить» неразрывным пробелом `~`. Это нужно делать в сокращениях («т.~е.», «г.~Петров»), размерных величинах («10~В/см»), а также в ссылках на номера страниц и объектов («с.~\pageref{p:SomePage}», «на рис.~\ref{f:SomeFigure}»).

Рассмотрим теперь, как можно принудительно разрывать строки. Команда перевода строки

```
\\[длина]
```

делает это, не заботясь о ее выравнивании. В ее параметре можно задать длину, которая добавляется к межстрочному интервалу. При положительной длине следующая строка отодвигается, а при отрицательной — приближается к разрываемой.

Разорвать строку, выровняв ее, позволяет команда

```
\linebreak[число].
```

При принудительном выравнивании строка может оказаться очень «рыхлой», поэтому параметр команды позволяет настроить это действие. Он задается числом, изменяющимся в пределах от 0 до 4, регулирующим «жесткость» указания компилятору от рекомендации (0), до обязательного выполнения (4). По умолчанию, используется значение 4, т. е. строка разрывается и выравнивается при любой заполненности.

При использовании параметра `draft` в стандартных классах строки с плохой заполненностью помечаются в конце небольшим прямоугольником.

2.2.2. Отступы

В начале абзаца, как правило, автоматически ставится небольшой пробел, называемый *отступом*. В стандартных классах первый абзац раздела делается без отступа, а в остальных абзацах он есть. В традициях русской полиграфии все абзацы начинаются отступом. Такую расстановку обеспечивает пакет `indentfirst`. Его нужно загружать в каждом документе, написанном по-русски. Регулировать наличие отступа позволяют команды `\indent` или `\noindent`. Они ставятся в самом начале абзаца. Первая из них вставляет, а вторая, наоборот, убирает отступ.

2.2.3. Пробелы

Компилятор устанавливает величины пробелов в зависимости от заполненности строки, добиваясь оптимальной плотности текста. Если же возникает необходимость вмешаться в его действия и вставить пробел нужной длины, это можно сделать несколькими способами.

Часто используемые пробелы определены в виде команд, приведенных в колонках табл. 2.3, выделенных фоном. Их длины, указанные в светлых колонках, установлены в относительных единицах, поэтому при изменении размера шрифта они изменяются согласованно с окружающим текстом.

Часто используемые пробелы

Таблица 2.3

<code>\quad</code>	2 em	<code>\enspace</code>	1/2 em	<code>\;</code>	1/4 em	<code>\,</code>	1/6 em
<code>\quad</code>	1 em	<code>\enskip</code>	1/2 em	<code>\:</code>	1/5 em	<code>\!</code>	-1/6 em

Вставить пробел любой длины позволяют команды

`\hspace{длина}` и `\hspace*{длина}`.

Если длина пробела положительна, следующий за ним текст, сдвигается вправо, а при отрицательной длине — влево. С помощью отрицательного пробела можно наложить одну часть

текста на другую, что не запрещено. Для примера, используя такое наложение, создадим новый символ:

$$=\hspace*{-.63em}/ \rightsquigarrow \neq.$$

TeX отбрасывает пробелы, находящиеся в начале и конце строки, в том числе заданные командой `\hspace` и командами из табл. 2.3, а вот команда `\hspace*` выполняется всегда. С ее помощью можно даже вылезти за левое поле, как показано в следующем примере:



```
← \[-1ex] \hspace*{-2em}\tiger[2em]\quad
```

Этот трюк выполнил набор команд: первая из которых переместила строку, сместив новую строку вверх,² вторая сдвинула ее начало на `-3em`, третья загрузила картинку размером `2em`,³ а четвертая отодвинула текст примера на левую границу страницы.

Пробел, отделяющий команду от последующего текста, «съедается» при компиляции. Восстановить его позволяет команда `\` . Например, «`1\AA` равен \rightsquigarrow `1\AA` равен» и «`1\AA\` равен \rightsquigarrow «`1\AA` равен». Ширину такого пробела компилятор регулирует сам.

2.2.4. Пружины

Вдобавок к пробелам имеются «пружины»

`\hfil` и `\hfill`,

способные заполнить все свободное пространство строки. Пружины являются пробелами, обладающими нулевой естественной длиной и бесконечной растяжимостью. L^AT_EX неявно использует их для выравнивания текста. Например, при разрыве строки командой `\` в ее конце ставится пружина `\hfil`, прижимающая текст влево. В стандартных конструкциях L^AT_EX, как правило, используются пружины `\hfil`, а команды `\hfill` позволяют пользователю корректировать выравнивание по своему усмотрению.

² Сдвиг компенсирует увеличение межстрочного интервала из-за большой высоты картинки.

³ Команда `\tiger[•]` определена как `\includegraphics[width=•]{tiger}`.

Пружина `\hfill`, имеющая большую жесткость, «сминает» менее жесткую пружину `\hfil` при совместном использовании.

Разберем действие пружин на следующих примерах.

Пример 1.

```
\hfil \tiger \hfil \tiger \\\
```



Здесь картинки расположены в строке несимметрично. Первая из них окружена пружинами `\hfil`, из которых левая, находящаяся в начале строки, игнорируется как обычный пробел. Вторая картинка находится на одинаковом расстоянии от правой границы текста и первой картинки, так как она окружена одинаковыми пружинами, одна из которых задана явно командой `\hfil`, а вторая неявно стоит в конце строки.

Пример 2.

```
\mbox{}\hfil \tiger \hfil \tiger \\\
```



В данном примере в начало строки поставлен пустой бокс `\mbox{}`, служащий «упором» для первой пружины. Как следствие, обе картинки, оказавшись зажатыми одинаковыми пружинами, распределились по строке симметрично.

Пример 3.

```
\mbox{}\hfil \tiger \hfill \tiger \\\
```



В последнем примере между картинками поставлена жесткая пружина `\hfill`, смявшая пружины `\hfil`, находящиеся в начале и конце строки, поэтому картинки разошлись по краям страницы.

Пара жестких пружин

`\dotfill`

и

`\hrulefill`

заполняют пробел точками или линией, например:

`|\dotfill|\hrulefill|\ \rightsquigarrow`

|.....|—————|

2.3. Выделение и выравнивание текста

Часть текста можно выделить подчеркиванием или отбивками, а также изменив шрифт или метод выравнивания. Использование шрифтов обсуждается в разд. 2.6.

Подчеркивание обеспечивает команда

`\underline{текст}`,

создающая бокс, в который помещается часть строки. Используя ее многократно, можно сделать многократное подчеркивание:

`\underline{\underline{пример}}` \rightsquigarrow пример.

Для выравнивания абзацев и больших массивов текста применяются команды и окружения, представленные в табл. 2.4. С их помощью текст можно центрировать, прижимать к одному из краев или выделять отбивками, т. е. дополнительными пустыми полями.

Таблица 2.4

Средства выравнивания массива текста

Действия	Окружения	Команды
Центрирование	<code>center</code>	<code>\centering</code>
Выравнивание слева	<code>flushleft</code>	<code>\raggedright</code>
Выравнивание справа	<code>flushright</code>	<code>\raggedleft</code>
Отбивка текста	<code>quote, quotation</code>	

При выравнивании текста с помощью команд и окружений, представленных в трех первых строках табл. 2.4, отключается таблица переноса слов, подавляются отступы и в каждую строку

вставляются пружины `\hfil`. Текст, выравниваемый по левому краю, поджимается пружинами справа. В тексте, выровненном по правому краю, пружины стоят в начале строк. Центрированный текст сжимается пружинами, стоящими по его краям.

Выполняя одну и ту же операцию, команды и окружения действуют на текст немного по-разному.

Сверху и снизу текста, выровненного окружениями `flushleft`, `flushright` и `center`, делается небольшая отбивка. Чтобы показать это, данный абзац помещен в окружение `flushleft`. Из примера видно, что слова не переносятся и правая граница текста остается «рваной».

Текст, выровненный командами `\centering`, `\raggedleft` и `\raggedright`, не отбивается. Чаще всего эти команды используются внутри окружений, например, `minipage` или `figure`.

Текст, помещаемый в окружения `quote` и `quotation`, имеет обычное выравнивание, но при этом со всех сторон он отбивается пустыми полями, хорошо выделяющими его на странице.

В окружении `quote` абзацы разделяются небольшой вертикальной отбивкой и не имеют отступа. Это видно в данном примере, в котором шрифт уменьшен для наглядности.

Выделение хорошо смотрится, при большом объеме текста, если его абзацы достаточно велики. Не стоит применять его для отдельных строк. Они будут выглядеть очень рыхлыми.

Чтобы показать разницу оформления текста, другой пример поместим в окружение `quotation`.

В окружении `quotation` абзацы начинаются с отступа, а отбивка между ними не делается. В данном примере также для наглядности используется уменьшенный шрифт.

Текст верстается в колонку, ширина которой меньше ширины строки окружающего текста. Как и в случае `quote`, для хорошего зрительного выделения абзацы должны содержать по несколько строк.

2.4. Вертикальное выравнивание

Из абзацев формируются страницы. Компилятор распределяет текст по страницам, используя алгоритмы вертикальной моды, в которой он оптимизирует промежутки между абзацами, формулами, заголовками, таблицами, рисунками и т. п. Единица вертикального выравнивания текста — *абзац*, но и внутри него расстояние между строками также может меняться.

Межстрочный интервал определяется произведением величин `\baselinestretch` и `\baselineskip`. Длина `\baselineskip` хранит высоту межстрочного интервала, зависящую от размера шрифта, а в команде `\baselinestretch` хранится десятичное число, дополнительно регулирующее расстояние между строками, по умолчанию равное единице. Если изменить его значение,⁴

```
\renewcommand\baselinestretch{1.5},
```

межстрочный интервал увеличится в полтора раза.

Как говорилось в разд. 2.2.1, расстояние между строками внутри абзаца позволяет регулировать команда перевода строки. Вариант «со звездочкой» данной команды:

```
\\*[длина],
```

разрывая строку, следит, чтобы следующая оказалась с ней на одной странице. При положительной длине следующая строка сдвигается вниз, а при отрицательной — вверх.

Команды вертикальных пробелов `\smallskip`, `\medskip` и `\bigskip` позволяют регулировать промежутки между абзацами. Обычно они ставятся в начале или в конце абзаца. Длины пробелов хранятся в командах `\smallskipamount`, `\medskipamount` и `\bigskipamount`.

Аналогично горизонтальным пробелам определены команды

```
\vspace{длина} и \vspace*{длина},
```

позволяющие вставить вертикальный промежуток любой высоты. Вертикальные пробелы, попадающие в начало или конец

⁴ Редактирование команд описано в разд. 6.1.

страницы, игнорируются, но сдвиг, заданный командой `\vspace*`, выполняется всегда.

Наряду с пробелами существуют и вертикальные пружины

`\vfil` и `\vfill`,

вставляемые между абзацами. Оказавшись внутри абзаца, пружина автоматически разбивает его в том месте, где она находится. Оказавшись же в начале или конце страницы, пружина, как и другие пробелы, игнорируется. По сравнению `\vfil` пружина `\vfill` обладает большей жесткостью.

В конце страницы \LaTeX неявно использует в пружину `\vfil`, прижимающую текст вверх. Это легко проверить, поставив в начале новой страницы конструкцию `\mbox{}\vfil`. В этом случае пустой бокс послужит упором для пружины, и набираемый текст, оказавшись между двумя одинаковыми пружинами, сдвинется в середину страницы. Аналогичная конструкция с командой `\vfill` приведет к тому, что текст опустится вниз, так как более жесткая пружина сомнет более мягкую. Проиллюстрируем сказанное на примере двух министраниц с вертикальными пружинами, стоящими в их начале:

`\mbox{}\vfil`

Размер страницы показан рамкой.

`\mbox{}\vfill`

Размер страницы показан рамкой.

Используя аналогии выравнивания строки пружинами `\hfil` и `\hfill`, легко построить нужный шаблон вертикального выравнивания текста на странице.

Регулировать разбиение текста на страницы помогают команды

`\newpage`
`\pagebreak[число]` и `\nopagebreak[число]`.

В вертикальной моде они служат аналогами команд перевода строки.

Команда `\newpage`, как следует из ее имени, должна начать новую страницу или новую колонку при многоколоночном наборе. Она не заботится о равномерности распределении текста по высоте разрываемой страницы. Если страница окажется слишком пустой, компилятор может проигнорировать ее выполнение.

Команда `\pagebreak` также завершает текущую страницу и начинает новую, а команда `\nopagebreak`, напротив, препятствует переходу на следующую страницу. Их параметр, изменяющийся в пределах 0–4, позволяет управлять жесткостью предписываемого компилятору указания от рекомендации (0), до обязательного выполнения (4), используемого по умолчанию. Команда `\nopagebreak` учитывается только при автоматическом разбиении текста на страницы, если же какая-то другая команда явно иницирует разрыв страницы, она игнорируется.

В отличие от команды `\newpage`, не заботящейся о выравнивании разрываемой страницы, `\pagebreak` может это сделать. Выравнивание зависит от действия деклараций

`\flushbottom` и `\raggedbottom`

Если активна декларация `\flushbottom`, страницы заполняются сверху до низу за счет увеличения промежутков между абзацами. Если выравнивание отменено декларацией `\raggedbottom`, внизу страниц остается пустое поле.

Эти декларации являются глобальными и их использование меняет режим верстки всех последующих страниц. В стандартных классах выравнивание `\flushbottom` активируется при использовании параметра `twoside`, чтобы обе страницы разворота были одинаковы. Для параметра `oneside` вертикальное выравнивание не используется.

Компилятор стремится не допустить переполнение страниц, поэтому команда `\nopagebreak` эффективна не всегда. В таких случаях можно изменить высоту текста страницы, используя команды

```
\enlargethispage{длина} ,
\enlargethispage*{длина} ,
```

увеличивающие его на указанную длину. Последняя дополнительно минимизирует имеющиеся на странице вертикальные пробелы.

В качестве примера использования команд, регулирующих вертикальное выравнивание, сделаем титульную страницу отчета, представленную на следующей странице:

```
\newpage\thispagestyle{empty}
\begin{center}
  МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
    ФЕДЕРАЦИИ \\\
  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
    ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ \\\
  Национальный исследовательский ядерный университет
    «МИФИ» \\\
  (НИЯУ МИФИ)

\vflll
  Отчет о научно-исследовательской работе \\\
  <<Современное состояние проблемы массы нейтрино>>
\vflll

\mbox{}\hfill
\begin{tabular}{lp{5em}r}
  Работу выполнил & & \\\
  магистр гр. ●-● & & И.И. Иванов\\[1ex]
  Руководитель & & \\\
  доц. кафедры № ● & & П.П. Петров
\end{tabular}

\vflll \today

\end{center}
\newpage
```

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
Национальный исследовательский ядерный университет
«МИФИ»
(НИЯУ МИФИ)

Отчет о научно-исследовательской работе
«Современное состояние проблемы массы нейтрино»

Работу выполнил
магистр гр. • - •

И. И. Иванов

Руководитель
доц. кафедры № •

П. П. Петров

25 апреля 2021 г.

Команды `\newpage`, стоящие в начале и в конце примера, ограничивают текст верстаемой страницы. Команда

```
\thispagestyle{стиль страницы}
```

с аргументом `empty` подавляет вывод ее номера. Для центрирования текста используется окружение `center`. Название института, заголовок отчета, сведения об исполнителе и его руководителе, а также дата создания отчета, выведенная командой `\today`, разделены пружинами `\vfill`, обеспечивающими их равномерное распределение по высоте страницы. Сведения об исполнителе и руководителе оформлены в виде таблицы, прижатой к правому краю страницы пружиной `\hfill`, упором для которой служит пустой бокс. Подписи исполнителя и руководителя разделены небольшим вертикальным пробелом путем раздвигания строк с помощью команды `\\[1ex]`.

2.5. Министраницы

Возможности верстки значительно расширяют министраницы. При создании министраницы достаточно объявить ее ширину, зная которую, компилятор верстает ее как обычную страницу, нарезая текст на строки заданной длины, помещает в бокс и встраивает его в текущую строку на общих основаниях.

Министраницы могут использоваться в любом месте программы, где может стоять хотя бы один текстовый символ. Они могут содержать сноски и практически любые объекты, за исключением плавающих (см. разд. 3.3), в которых они используются довольно часто для компоновки рисунков и подписей.

Министраницу можно создать с помощью окружения:

```
\begin{minipage}[привязка][высота][выравнивание]{ширина}
```

которое имеет аргумент, задающий ширину министраницы, и набор параметров, определяющих ее высоту, а также методы встраивания в строку и распределения текста по высоте.

Привязка к строке задается следующими идентификаторами:

- c — министраница центрируется относительно базисной линии строки, в которую она встраивается (этот параметр используется по умолчанию);
- t — верхняя строка министраницы встраивается в строку;
- b — в строку встраивается нижняя строка министраницы.

Если задана высота министраницы, текст можно распределить по вертикали с помощью идентификаторов c, b, t или s:

- c — текст центрируется;
- b — текст прижимается к нижнему краю (по умолчанию);
- t — текст прижимается к верхнему краю;
- s — текст равномерно распределяется по высоте министраницы.

Пример использования министраниц для компоновки рисунков и подписей приведен далее.

Аналогом окружения `minipage` является команда:

```
\parbox[привязка]{высота}[выравнивание]{ширина}{текст},
```

имеющая такой же синтаксис и тоже создающая министраницу. Ее удобно использовать для небольших фрагментов текста.

2.6. Текстовые шрифты

Важной составляющей верстки является шрифтовое оформление документа. Следует учитывать, чем реже меняется шрифт, тем легче читается текст, а хорошо подобранные шрифты названий разделов, подписей, сносок и т. д. выделяют их из основной массы текста и делают наглядной структуру документа.

По умолчанию, стандартные классы \LaTeX используют шрифты Computer Modern.

Текстовый шрифт имеет следующие параметры: семейство (гарнитура), контрастность, начертание, размер (кегель) и кодировку. Любой из перечисленных параметров можно изменять

независимо от других, при этом последовательность изменения нескольких параметров не влияет на конечный результат.

2.6.1. Гарнитурa Computer Modern

Гарнитура — совокупность шрифтов разной насыщенности, с различным начертанием и размером, принадлежащих одному семейству. Шрифты Computer Modern содержат три гарнитурa: Roman — основной шрифт данного документа, SansSerif — шрифт, которым набран этот фрагмент, и TypeWriter — шрифт печатной машинки, представленный данным фрагментом.

Контрастность шрифта определяют ширина его символов и насыщенность, зависящая от их «толщины». Шрифты бывают светлые (тонкие), нормальные, жирные и т. д. Качественные типографские шрифты имеют большой диапазон контрастностей. Гарнитурa Computer Modern содержат нормальные и уширенные **жирные шрифты**.

Начертание шрифта задает стиль формы символов. В шрифтах Computer Modern имеется обычное начертание, *наклонный шрифт, курсив, или italic*, и КАПИТЕЛЬ, в КОТОРОЙ СТРОЧНЫЕ БУКВЫ ИМЕЮТ ВИД УМЕНЬШЕННЫХ ПРОПИСНЫХ. В описаниях команд использовался прямой курсив гарнитурa Roman, к сожалению, недоступный в обычном интерфейсе работы со шрифтами.

Гарнитура Roman относится к категории пропорциональных шрифтов «с засечками». Она наиболее богата шрифтами, которые имеют нормальную и жирную насыщенность всех перечисленных начертаний:

Обычный шрифт (upright).

Курсив (italic).

Прямой курсив (upright italic).

Наклонный шрифт (slanted).

ПРОПИСНОЙ ШРИФТ ИЛИ КАПИТЕЛЬ (SMALL CAPITALS).

Жирный шрифт (bold).

Жирный курсив (bold italic).

Жирный наклонный шрифт (slanted bold).

ЖИРНАЯ КАПИТЕЛЬ (BOLD SMALL CAPITALS).

Гарнитура SansSerif относится к категории пропорциональных рубленных шрифтов. Она также содержит шрифт нормальной насыщенности и жирный шрифт. Начертания наклонного шрифта и курсива совпадают, капитель подставляется из гарнитуры Roman, а жирная капитель отсутствует:

Пример обычного шрифта.

Пример наклонного шрифта и курсива (slanted ≡ italic).

ПРИМЕР ПРОПИСНОГО ШРИФТА.

Пример жирного шрифта.

Пример жирного курсива (slanted ≡ italic).

Гарнитура TypeWriter относится к категории шрифтов с «засечками» и фиксированной шириной символов. В ней имеются шрифты нормальной насыщенности всех начертаний, а жирные шрифты отсутствуют. В тексте, набранном в этой гарнитуре, автоматический перенос слов не используется, пробелы имеют фиксированную ширину, а разбивку на строки зачастую приходится регулировать вручную командами `\\`.

Пример обычного шрифта.

Пример курсива.

Пример наклонного шрифта.

ПРИМЕР КАПИТЕЛИ.

Для изменения параметров шрифта имеется целый ряд команд, деклараций и окружений, представленных в табл. 2.5. Команды предназначены для оформления небольших фрагментов текста в пределах абзаца, а окружения — для больших массивов текста. Декларации удобно применять в ячейках таблиц, заголовках, подписях и т. д., которые, являясь группой, автоматически ограничивают область изменения шрифта.

Как уже отмечалось, параметры шрифта могут комбинироваться в любом порядке, поэтому последовательности команд `\textit{\textbf{•}}` и `\textbf{\textit{•}}`, производящие *жирный курсив*, являются эквивалентными.

Таблица 2.5

Параметры гарнитур Computer Modern

Команды	Декларации	Окружения	Примеры гарнитур		
<code>\textrm{●}</code>	<code>\rmfamily</code>	<code>rmfamily</code>	Roman	Serif	Typewriter
<code>\textsf{●}</code>	<code>\sffamily</code>	<code>sffamily</code>			
<code>\texttt{●}</code>	<code>\ttfamily</code>	<code>ttfamily</code>			
<code>\textmd{●}</code>	<code>\mdseries</code>	<code>mdseries</code>	Medium	Medium	Medium
<code>\textup{●}</code>	<code>\upshape</code>	<code>upshape</code>	Upright	Upright	Upright
<code>\textit{●}</code>	<code>\itshape</code>	<code>itshape</code>	<i>Italic</i>	<i>Italic</i> ≡ <i>Slanted</i>	<i>Italic</i>
<code>\textsl{●}</code>	<code>\slshape</code>	<code>slshape</code>	<i>Slanted</i>	<i>Slanted</i> ≡ <i>Italic</i>	<i>Slanted</i>
<code>\textsc{●}</code>	<code>\scshape</code>	<code>scshape</code>	SMALLCAPITALS	SMALLCAPITALS	SMALLCAPITALS
<code>\textbf{●}</code>	<code>\bfseries</code>	<code>bfseries</code>	Bold	Bold	Bold
<code>\textup{●}</code>	<code>\upshape</code>	<code>upshape</code>	Bold	Bold	Bold
<code>\textit{●}</code>	<code>\itshape</code>	<code>itshape</code>	<i>BoldItalic</i>	<i>BoldItalic</i>	<i>BoldItalic</i>
<code>\textsl{●}</code>	<code>\slshape</code>	<code>slshape</code>	<i>BoldSlanted</i>	<i>BoldSlanted</i>	<i>BoldSlanted</i>
<code>\textsc{●}</code>	<code>\scshape</code>	<code>scshape</code>	BOLDsmallcap	BoldSmallCap	BoldSmallCap
<code>\emph{●}</code>	<code>\em</code>		<i>Italic</i> ⇔ Upright	<i>Italic</i> ⇔ Upright	<i>Ital.</i> ⇔ Upr.

Последовательность команд `\texttt{\textit{\textbf{●}}}` даст *обычный курсив шрифта Typewriter*, так как в данной гарнитуре нет жирных шрифтов. Если требуемое начертание или контрастность отсутствуют в гарнитуре, замена шрифта не приведет к ошибке компиляции, но часть его параметров останется неизменной.

Команда `\emph{●}` и декларация `\em` предназначены для переключения с прямого шрифта на курсив, и наоборот, в зависимости от того, какое начертание имел предшествующий текст. Это иллюстрирует `\emph{данный \em пример}` \rightsquigarrow *данный пример*.

Декларация `\normalfont` восстанавливает семейство, контрастность и начертание, установленные по умолчанию для основного шрифта документа.

Конструкции, изменяющие несколько параметров шрифта, оказываются довольно громоздкими, поэтому для быстрого доступа к часто используемым шрифтам определены специальные декларации:

<code>\sf</code>	\equiv <code>\sffamily\mdseries\upshape</code>	Стандартный Serif.
<code>\tt</code>	\equiv <code>\ttfamily\mdseries\upshape</code>	Стандартный TypeWriter
<code>\rm</code>	\equiv <code>\rmfamily\mdseries\upshape</code>	Стандартный Roman.
<code>\it</code>	\equiv <code>\rmfamily\mdseries\itshape</code>	<i>Курсив Roman.</i>
<code>\sl</code>	\equiv <code>\rmfamily\mdseries\slshape</code>	<i>Наклонный Roman.</i>
<code>\sc</code>	\equiv <code>\rmfamily\mdseries\scshape</code>	КАПИТЕЛЬ ROMAN.
<code>\bf</code>	\equiv <code>\rmfamily\bfseries\upshape</code>	Жирный Roman.

Покажем их действие, изменяя оформление заголовка раздела:

<code>\section*{Заголовок}</code>	\rightsquigarrow	Заголовок
<code>\section*{\sf Заголовок}</code>	\rightsquigarrow	Заголовок
<code>\section*{\it Заголовок}</code>	\rightsquigarrow	<i>Заголовок</i>
<code>\section*{\normalfont Заголовок}</code>	\rightsquigarrow	Заголовок
<code>\section*{\small Заголовок}</code>	\rightsquigarrow	Заголовок

Предваряя следующий раздел, в последнем примере, мы изменили размер шрифта заголовка.

В большинстве случаев основной шрифт документа принадлежит семейству Roman, однако если основным семейством является Serif, в расшифровке команд `\it`, `\sl`, `\sc` и `\bf` команда `\rmfamily` заменяется на `\sffamily`. В этом случае курсив и каллиграф соответствуют наклонному и обычному шрифту:

`\it` \equiv `\sl` \rightsquigarrow *наклонный шрифт*, `\sc` \equiv `\sf` \rightsquigarrow *обычный шрифт*.

2.6.2. Размеры шрифтов

Размеры шрифтов различных составляющих частей документа отсчитываются от размера его основного шрифта, который обычно устанавливается при загрузке класса документа. В стандартных классах определены три размера основного шрифта: 10pt, 11pt и 12pt. Коллекция пакетов `extsizes` добавляет к ним размеры 8pt, 9pt, 14pt, 17pt и 20pt. Чтобы их использовать, нужно загрузить класс документа, начинающийся с префикса `ext`: `extarticle`, `extbook`, `extletter`, `extproc`, `extreport`.

Стандартные размеры шрифтов⁵

Таблица 2.6

Декларации и окружения	Примеры	10pt	11pt	12pt
<code>\tiny</code>	Abcdefghijkl	5/6	6/7	6/7
<code>\scriptsize</code>	Abcdefghij	7/8	8/9.5	8/9.5
<code>\footnotesize</code>	Abcdefghi	8/9.5	9/11	10/12
<code>\small</code>	Abcdefgh	9/11	10/12	11/13
<code>\normalsize</code>	Abcdefg	10/12	11/13.6	12/14.5
<code>\large</code>	Abcdefg	12/14	12/14	14/18
<code>\Large</code>	Abcdef	14/18	14/18	17/22
<code>\LARGE</code>	Abcd	17/22	17/22	20/25
<code>\huge</code>	Abcd	20/25	20/25	25/30
<code>\Huge</code>	Abc	25/30	25/30	25/30

⁴ Пары значений (размер шрифта)/(межстрочный интервал) указаны в единицах pt. Примеры приведены для основного размера 11pt.

Изменение размера шрифта в тексте регулируют декларации, перечисленные в табл. 2.6. Наряду с декларациями имеются одноименные окружения.

Область действия деклараций ограничена группой, поэтому изменение кегля внутри группы не коснется текста за ее пределами. Напомним, что основные компоненты документа, такие как аннотация, заголовки, подписи, сноски и т. д. образуют группы.

Чтобы тест не был слишком разреженным или плотным, в окружениях при смене кегля меняется также и межстрочный интервал. При использовании деклараций меняется только кегль, а для изменения межстрочного интервала текст следует поместить в министраницу (см. разд. 2.5).

Пример уменьшения шрифта заголовка приведен ранее, а здесь скажем всем разработчикам L^AT_EX

```
{\Huge <<огромное спасибо>>!} ~\n
```

«огромное спасибо»!

2.7. Кодировка текста

Компилятор `pdftex` работает в 8-битной кодировке, в которой только 256 символов можно идентифицировать кодами, тем не менее количество используемых символов не ограничено, так как большинство из них задаются командами.

Пакет `inputenc` преобразует буквы национальных алфавитов в команды, например: æ ~\ae, или Б ~\IeC {\CYRB}. Он загружает кодировку, связывающую коды символов с соответствующими командами:

```
\usepackage[кодировка]{inputenc}.
```

Загружаемая кодировка должна соответствовать кодировке файлов программы, иначе скомпилированный документ будет содержать абракадабру.

Полный набор символов европейских алфавитов обеспечивает кодировка `utf8`. Она также используется в качестве рабочей

кодировки оболочкой **TeXStudio** и многими другими редакторами, поэтому файлы программы не требуют перекодировки при чтении и записи. Все это делает ее наиболее удобной и часто используемой.

```
\usepackage[utf8]{inputenc}
\usepackage{cmap}
\usepackage{textcomp}
\usepackage[russian]{babel}
```

Для набора текста достаточно загрузить кодировку **utf8**, пакеты **cmap**, **textcomp**, а также настройки русского языка, как показано выше. В этом случае будут доступны представленные в табл. 2.7 и Б.2 символы стандартного пакета **textcomp** и большая часть букв алфавитов латиницы и кириллицы.

Таблица 2.7

Текстовые символы кодировки utf8

/	\	×	±	÷	*	µ	№	Ω	∪
<	>	°	°C	*	·	...	○	●	◯
←	→	†	‡	#	%	% ₀₀	% ₀₀₀	¶	§
↑	↓	˘	ˆ	˙	˚	?			┌
<	>	á	o	1	2	3	½	¼	¾
«	»	©	®	℠	SM	TM	€	℞	♪
,	„	¢	đ	f	£	£	ℕ	P	W
‘	’	©	¥	€	¤	%	—	—	—
“	”	..	’	”	—	ˆ	˘	˙	˚

Наведите курсор на символ, чтобы открыть подсказку.

Таблица 2.8

Диакритические знаки

à	á	ä	â	ã	ā	á	à	â	á	â	ã	ä	å	ä	å

Наведите курсор на символ, чтобы открыть подсказку.

Алфавиты, основанные на латинице, в основном состоят из латинских букв и их акцентированных вариантов. Часть букв

(например, \ss ~> ß) задана командами, акцентированные буквы также генерируют команды, указанные в табл. 2.8.

Таблица 2.9

Буквы алфавитов латиницы

Àà	Áá	Ââ	Ãã	Ää	Åå	Āā	Ăă	Ąą	Ȧȧ
Ææ	Āā	Ĕĕ	Ċċ	Ĉĉ	Ċċ	Čč	Çç	Ðð	Ďď
Đđ	Èè	Éé	Êê	Ëë	Ēē	Ěě	Ěě	Ěě	Ěě
Ĝĝ	Ğğ	Ġġ	Ģģ	Ĥĥ	Ĝĝ	Ģģ	Ĥĥ	Ĥĥ	İi
Îî	Ïï	Īī	Ĭĭ	Įį	Ĭĭ	Ĭĭ	Ĭĭ	Ĭĭ	Ĭĭ
Ĵĵ	Ĵĵ	Ķķ	Ļļ	Ĺĺ	Ĺĺ	Ĺĺ	Ññ	Ńń	Ňň
Ŋŋ	Ŋŋ	Òò	Óó	Ôô	Õõ	Öö	Øø	Ōō	Ŏö
Œœ	Œœ	Œœ	Œœ	Œœ	Œœ	Œœ	Œœ	Œœ	Œœ
Šš	Šš	Šš	Šš	Šš	Šš	Šš	Šš	Šš	Šš
Ŧŧ	Ŧŧ	Ŧŧ	Ŧŧ	Ŧŧ	Ŧŧ	Ŧŧ	Ŧŧ	Ŧŧ	Ŧŧ
Ũũ	Ũũ	Ũũ	Ũũ	Ũũ	Ũũ	Ũũ	Ũũ	Ũũ	Ũũ
		Ŷŷ	Ŷŷ	Ŷŷ	Ŷŷ	Ŷŷ	Ŷŷ	Ŷŷ	Ŷŷ
		ŸŹ	ŸŹ	ŸŹ	ŸŹ	ŸŹ	ŸŹ	ŸŹ	ŸŹ

Наведите курсор на символ, чтобы открыть подсказку.

Таблица 2.10

Буквы алфавитов кириллицы

Аа	Ăă	Ää	Ææ	Бб	Вв	Гг	Íí	Гг
Гг	Ђђ	Гг	Ґґ	Дд	Ее	Èè	Ёё	Ёё
Жж	Жж	Жж	Жж	Зз	Зз	Зз	Ии	Йй
Йй	Йй	Йй	Кк	Кк	Кк	Кк	Кк	Кк
Ққ	Лл	Лл	Лл	Мм	Мм	Нн	Нн	Нн
Нн	Нн	Оо	Öö	Өө	Пп	Пп	Рр	Рр
Сс	Сс	Тт	Тт	Цц	Уу	Уу	Уу	Уу
Уу	Фф	Хх	Хх	Хх	Хх	Цц	Чч	Чч
Чч	Чч	Чч	Шш	Шш	Ъъ	Ыы	Ыы	Ыы
Ъъ	Ээ	Ээ	Єє	Юю	Яя	Ии	Йй	Јј
Ѕѕ	Уу	Уу	Һһ	Ҥҥ	Ҥҥ	Цц	Әә	Әә
Ққ	Ғғ	Ғғ	Ҝҝ	Ҝҝ	Ҝҝ	Ҝҝ	Ҝҝ	Ҝҝ

Наведите курсор на символ, чтобы открыть подсказку.

Буквы латиницы и кириллицы представлены в табл. 2.9, Б.4 и 2.10, Б.5. В табл. 2.9 и 2.10 подчеркиванием выделены буквы, недоступные в кодировках T2A и T1, загружаемых по умолчанию для документов, написанных по-русски. Подробнее о кодировках шрифтов можно узнать в книгах [3, 4] и прил. Б.

Команды, генерирующие символы и буквы, не показаны в табл. 2.7–2.10, чтобы не загромождать их. Меню с аналогичными таблицами, предназначенные для ввода команд или непосредственно символов, можно найти в левой панели окна *TeXStudio*. Прил. Б содержит таблицы, в которых символы и буквы приведены вместе командами и *unicod*e-кодами.

Буквы алфавитов кириллицы отличаются большим разнообразием. Каждую из них пакет *inputenc* заменяет при компиляции командой `\IeC{CYR• }`. Например, имя «Петр» преобразуется к виду `\IeC{CYRP } \IeC{cyre } \IeC{cyrt } \IeC{cyrr }`.

Начиная с 2020 г., текст направляемый в служебные файлы, генерируемые в процессе компиляции, в команды не переводится, что открывает возможность использования русских букв в метках и ярлыках.

Как правило, текст документа, генерируемого L^AT_EX в кодировке *utf8*, можно копировать в другие документы и приложения, однако в программе *AdobeReader* с этим могут возникнуть проблемы. Чтобы избежать их, пакет *cmapp* вставляет в pdf-документ кодировочную таблицу.

Фрагменты текста, копируемые в рукопись из других приложений, могут содержать неопределенные *unicode*-символы, вызывающие ошибку компиляции:

```
! Package inputenc Error: Unicode character •(U+####)
(inputenc) not set up for use with LaTeX.
```

При ее появлении следует ввести такой символ в поле комментария (или после `\end{document}`), а затем заменить его повсюду соответствующей командой L^AT_EX. В *TeXStudio* для ввода можно воспользоваться меню «Edit ⇒ Insert Unicode Character», скопировав код символа из сообщения об ошибке во всплывающее окно.

2.8. Тире, кавычки, точки, запятые

В \LaTeX определено большое количество различных кавычек, которые собраны в двух левых колонках в нижней части табл. 2.7. Их можно вводить с помощью команд и с клавиатуры, как обычные символы. Для комфортного набора кавычек в шрифтах также предусмотрены лигатуры. *Лигатура* — преобразование последовательности из нескольких символов в общий символ. Сравните, например, последовательность букв $f\{f\}1 \rightsquigarrow ffl$ с их лигатурой ffl .

Две запятые дают лигатуру нижней кавычки (`,,` \rightsquigarrow `„`), две верхние левые и правые кавычки — двойные верхние кавычки (`''` \rightsquigarrow `”` и `““` \rightsquigarrow `“`).

В англоязычной литературе используются одинарные и двойные верхние кавычки, при этом одинарные «вкладываются» в двойные: `“To be, or not to be?” — that is the question.`

В русской полиграфии используются „лапки“ и «елочки», для которых также есть лигатуры: `<<` \rightsquigarrow `«`, `>>` \rightsquigarrow `»`. «Следует помнить, что основными кавычками являются „елочки“, а „лапки“, как правило, служат в качестве вложенных!».

Для елочек пакет `babel`, загружаемый с параметром `russian`, вводит дополнительные команды: `"<` \rightsquigarrow `«` и `">` \rightsquigarrow `»`.

Дефис, используемый для разделения слов (например, *темнокрасный*), вводится обычным образом с клавиатуры. Для ввода тире имеются команды и лигатуры.

Короткое тире, применяемое для разделения диапазона цифр, вводится с помощью лигатуры `--` или команды `\textendash`. Например, `7--8` \rightsquigarrow `7–8`, а лучше `7\,--\,8` \rightsquigarrow `7–8`. Тот же диапазон, помещенный в формулу, выглядит иначе: `$7-8$` \rightsquigarrow `7 – 8`.

Длинное тире, используемое в сложных предложениях, вводится с помощью лигатуры `---` или команды `\textemdash`:

`<<Зачем тебе жужжать, если ты не пчела?>> ---` подумал Винни Пух. \rightsquigarrow «Зачем тебе жужжать, если ты не пчела?» — подумал Винни Пух.

Для русского языка пакет `babel` определяет целых три типа длинных тире. Одно из них, `--~` \rightsquigarrow `|—|`, эквивалентно лигатуре `---` или команде `\textemdash`. Как видно из примера, оно не имеет отбивок, а два других выводятся с небольшой правой отбивкой:

`"---|` \rightsquigarrow `|—|` и `"---*|` \rightsquigarrow `|—|`.

Последний вариант рекомендован для использования в прямой речи.

Для набора многоточия применяется команда

`\dots`,

которая определена как в тексте, так и в формулах, например `\pi=3.14\dots` \rightsquigarrow $\pi = 3.14\dots$ и т. п. . .

В англоязычных странах разделителем целой и дробной частей числа служит десятичная точка. В российской полиграфии для этого чаще используется запятая, однако это не является обязательным требованием. В документах, верстаемых в `LATEX`, следует пользоваться десятичной точкой, так как в формулах запятая автоматически отделяется небольшим пробелом, разбивающим число на две части. Сравните $\pi = 3.14159\dots$, $e = 2.71828\dots$ и $\pi = 3,14159\dots$, $e = 2,71828\dots$. Из-за этого зарубежные издательства и редакции переводных российских журналов не принимают рукописи с десятичной запятой.

2.9. Вывод текста программы

Для вывода небольших фрагментов программы, содержащих команды и служебные символы, определены команды

`\verb • текст •` и `\verb* • текст •`,

имеющие особый синтаксис. На месте кружков, служащих для выделения аргумента команд, можно поставить любой символ, отсутствующий в тексте. Аргумент (`текст`) должен уместиться в пределах одной строки. Он выводится как единое целое «шрифтом пишущей машинки», при этом `\verb*` отличается тем, что

заменяет пробелы символом пробела `\` . В качестве примера приведем вывод кода строчной формулы:

```
\verb!$y=\alpha x$!  $\leadsto$  $y=\alpha x$  $\leadsto$   $y = \alpha x$ .
```

Для вывода нескольких строк программы имеются окружения

`verbatim`

и

`verbatim*`,

воспроизводящие текст именно так, как он набран, включая пробелы и разбиение на строки. Как и в случае команд, вариант со звездочкой отличается тем, что использует символ `\` для печати пробелов. Пример использования окружения `verbatim` приведен на с. 72.

Перечисленные команды и окружения не могут находиться в аргументах и параметрах других команд и окружений, а также в ячейках таблиц.

Глава 3

Рисунки и таблицы

Рисунки и таблицы состоят из непосредственно иллюстрации или таблицы и подписи к ним. Каждая составляющая обрабатывается отдельно по своим правилам.

Положение рисунков и таблиц в документе редко соответствует порядку их следования в программе. Как правило, они помещаются вверху или внизу страницы, или выносятся на специальные страницы, содержащие только рисунки и таблицы. Их положение меняется, в зависимости от заполненности страниц, они как бы «плавают» в документе и потому называются *плавающими объектами*.

В данном разделе рассматриваются ресурсы, которыми располагает L^AT_EX для работы с иллюстрациями, затем описываются средства, позволяющие формировать таблицы, а в конце обсуждается компоновка плавающих объектов и средства контроля их положения в документе.

3.1. Иллюстрации

Иллюстрации загружаются из графических файлов, обычно находящихся в рабочей папке вместе с документом. Если иллюстраций много, для них можно создать отдельную папку.

Компилятор считывает из файла информацию о размерах картинка, если необходимо, преобразует ее и вычисляет новый размер. Если же нужной информации в файле нет, он выдаст сообщение об ошибке. Размер задают координаты нижнего левого и правого верхнего угла прямоугольника, в который вписана картинка. Как правило, они хранятся в специальном разделе файла, называемом `BoundingBox`.

В pdf-документ иллюстрация вставляется непосредственно при компиляции. В остальных случаях она замещается пустым боксом вычисленного размера, адресом файла, содержащего иллюстрацию, и описанием последовательности преобразований графических данных. Тогда для генерации документа дополнительно используется постпроцессор, например программа `dvips` (см. рис. 1.2), встраивающий графику в документ.

3.1.1. Форматы графики

Форматы графики зависят от формата генерируемого документа. В PostScript-документы можно вставить рисунки в формате PostScript (файлы с расширением `•.ps` и `•.eps`), а также в форматах растровых изображений Windows Bitmap и PCExchange (файлы с расширением `•.bmp`, `•.pcx`). Документы с расширением pdf могут содержать графику в форматах Portable Document Format (`•.pdf`), Joint Photographic Experts Group (`•.jpg` и `•.jpeg`), Portable Network Graphics (`•.png`) и Lossy/Lossless Coding of Bi-Level Images (`•.jbig2` и `•.jb2`).

Универсальным форматом, поддерживающим как векторную, так растровую графику является PostScript и его подмножество pdf. Для генерации иллюстраций программами, не имеющими вывода в PostScript и pdf, в ОС Windows10 можно воспользоваться следующим методом.

- Установите PostScript драйвер. Для этого откройте меню настроек системы «Принтеры и сканеры ⇒ Добавить принтер или сканер», затем в меню «Установка принтера» выберите

«Добавить локальный или сетевой принтер с параметрами, заданными вручную». Выбрав порт «Используйте существующий порт», установите для него «FILE:(Печать в файл)». Затем в меню «Установка драйвера принтера» выберите «Изготовитель: Microsoft» и «Принтеры: Microsoft PS Class Driver». В завершении запретите общий доступ к данному принтеру.

- Создайте иллюстрацию в любом приложении и распечатайте ее в файл с помощью данного драйвера. Получится рисунок, окруженный пустыми полями. Его размер соответствует границам страницы, на которой он «напечатан».
- Чтобы обрезать поля, откройте файл в программе GhostView и сделайте преобразование ps \rightsquigarrow eps. Для этого откройте меню File \Rightarrow PS to EPS, включите в появившемся окне флаг автоматического определения BoundingBox и сохраните новый файл с расширением eps. Открыв его в программе GhostView и убедившись, что поля обрезаны,¹ удалите начальный файл.
- Выполнив указанные действия, получите PostScript-рисунок, который непосредственно во время компиляции «на лету» преобразуется в формат pdf. Чтобы сгенерировать pdf-файл вручную, в папке с рисунком откройте командную консоль и выполните команду
`epstopdf <имя файла>`,
указав имя PostScript-файла.
- К сожалению, из-за ошибки драйвера, устанавливаемого в Windows10, pdf-рисунок вновь оказывается окружен пустыми полями. Ниже обсуждается метод их обрезания средствами L^AT_EX.

¹ Чтобы видеть границы рисунка, включите флаг Show Bounding Box в меню Options.

В ОС Linux для генерации PostScript-рисунка специальных драйверов не требуется. Скорректировать `BoundingBox` позволяет программа `ps2eps`, которая запускается командной строкой:

```
ps2eps -B -g -f <имя файла>.
```

Данные ключи иницируют вычисление `BoundingBox`. Программа

```
epstopdf <имя файла>
```

используется для преобразования PostScript \rightsquigarrow pdf.

3.1.2. Обработка иллюстраций

Для работы с графикой необходимо загрузить пакет `graphics` из коллекции `latex-graphics` [13], определяющий команды

```
\includegraphics[параметры]{имя файла},  
\includegraphics*[параметры]{имя файла}.
```

Обе они вставляют иллюстрации и различаются лишь тем, что первая выводит содержимое графического файла целиком, а вторая — только часть, находящуюся в пределах `BoundingBox`.

Нижняя граница `BoundingBox` иллюстрации выравнивается с базисной линией строки, в которую она встраивается.

В аргументе команд указывается имя файла, находящегося в рабочей папке, или путь к нему. Путь может быть абсолютным и относительным. Последний указывается относительно рабочей папки, например `figures/file`, если рисунки собраны в папку `figures`. Команда

```
\graphicspath{список папок}
```

позволяет задать список папок, в которых компилятор будет искать рисунки. Для этих рисунков указывается только имя файла. Папки списка заключаются в фигурные скобки, например

```
\graphicspath{{/eps}{/pdf}}.
```

Расширение имени файла лучше опустить, тогда компилятор сам будет искать файл допустимого графического формата. При верстке PostScript-документов поиск идет среди файлов с расширениями `•.bmp`, `•.psx`, `•.eps` или `•.ps`, а для pdf-документов — `•.pdf`,

•.jpg, •.jpeg, •.png и •.jbig2 или •.jb2. Если в папке имеются файлы `picture.eps` и `picture.pdf`, команда `\includegraphics{picture}` использует первый в PostScript, а в второй — в pdf-документе. Использование же имени `picture.eps` при генерации pdf-документа даст ошибку, так как форматы иллюстрации и документа не будут соответствовать друг другу.

Параметры команды `\includegraphics` используются для обработки иллюстрации. Они перечисляются через запятую. Очередность их следования определяет последовательность преобразований картинки. Полный перечень параметров приведен в [3], перечислим лишь те, что позволяют вращать и масштабировать иллюстрации, а также корректировать их границы.

`scale` — масштабирование относительно размеров `BoundingBox`.

Например, `scale=0.7` означает уменьшение до 70 % натурального размера иллюстрации.

`width` — масштабирование до заданной ширины. Например, `width=.75\linewidth` — до 75 % длины строки.

`height` — масштабирование до заданной высоты. Например, `height=.4\textheight` — до 40 % высоты страницы.

`angle` — поворот на угол, заданный в градусах, например `angle=7`. Положительные и отрицательные значения соответствуют вращению по часовой стрелке и против часовой стрелки.

`origin` — положение центра вращения, заданное координатами, привязанными к `BoundingBox` и/или к базисной линии строки. Значение по умолчанию: `origin=lb`. Картинка вращается относительно своего левого нижнего угла, а при значении `origin=c` — относительно своего центра. Первая буква — горизонтальная координата, которая может принимать значения `l`, `r` и `c`, что соответствует левой, правой границе и середине ширины `BoundingBox`. Вторая буква — вертикальная координата. Ее значения `t` и `b` привязывают центр вращения к верхней и нижней границе `BoundingBox`, `c` — к середине его высоты, а `B` — к положению базисной линии.

`trim` — корректировка `BoundingBox`. Указываются величины смещений левой, нижней, правой и верхней границ картинки. Положительные значения соответствуют обрезанию, а отрицательные — увеличению размеров. Например,

```
trim= 1mm 0 0 -4mm,
```

означает обрезание левой и увеличение верхней границ картинки. Если числа указаны без размерности, по умолчанию используется большой пункт `bp`, равный 0.3528 мм.

`clip` — обрезание картинки по границам `BoundingBox`. Команда `\includegraphics*` автоматически использует этот параметр.

Если используется один из параметров `height`, или `width`, размеры иллюстрации изменяются пропорционально, а при их совместном использовании размеры подгоняются до заданных величин без сохранения пропорциональности.

Параметры `trim` и `clip` позволяют внести коррекцию, если графический файл сделан неправильно, или вырезать из него часть, которую необходимо вывести в документ. При этом, используя команду

```
\fbox{текст} ,
```

можно поместить иллюстрацию в рамку, которая позволит увидеть как размеры `BoundingBox`, так и положение картинки в отведенном для нее прямоугольнике. Однако прежде нужно обнулить длину² `\fboxsep`, задающую ширину пустого поля между рамкой и текстом.

Разберем пример корректировки границ картинки, окруженной пустыми полями. Он показан на рис. 3.1 вместе с командами, формирующими этот рисунок.

Вверху показана исходная иллюстрация, на которой обозначены углы пустых полей. Для наглядности она помещена в рамку и отделена от остальных картинок пробелом высотой в одну строку.

² Смотри изменение длин на с. 39.

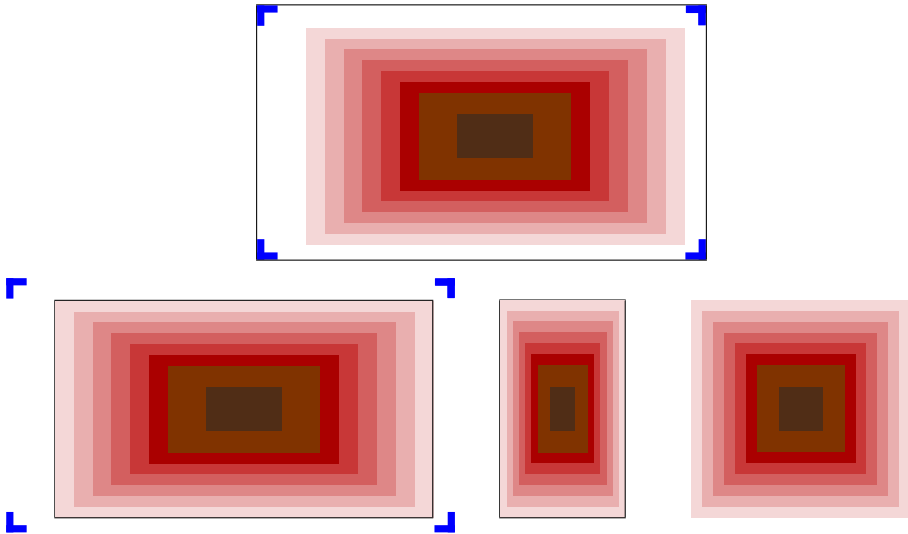


Рис. 3.1. Обработка иллюстраций с некорректными границами:

```

\begin{figure}
  \centering
  \setlength\fbboxsep{0pt}
  \fbbox{\includegraphics{BoundingBox}}\[\baselineskip]
  \fbbox{\includegraphics
    [trim= 6.5mm 2mm 3mm 3mm]{BoundingBox}}
  \hfill
  \fbbox{\includegraphics
    [trim= 6.5mm 2mm 3mm 3mm, scale=.575, angle=90,
    clip]{BoundingBox}}
  \hfill
  \includegraphics*
    [trim= 6.5mm 2mm 3mm 3mm, width=7.5em, height=7.5em]
    {BoundingBox}
  \caption{Обработка иллюстраций с некорректными
    границами.}
  \label{f:BoundingBox}
\end{figure}

```


Та же иллюстрация выведена внизу в трех видах. Между картинками с помощью пружин `\hfill` вставлены зазоры.

После коррекции `BoundingBox` с помощью параметра `trim` размеры левой картинке соответствуют размерам раскрашенного прямоугольника, о чем свидетельствует рамка вокруг него. Однако поля не обрезаны, потому картинка выводится целиком вместе с пустыми полями, углы которых показаны стрелками, вылезшими за левую границу страницы.

Поля двух других картинок обрезаются. После коррекции `BoundingBox` они масштабируются: центральная пропорционально, а правая с помощью параметров `width` и `height` преобразуется в квадрат. Центральная картинка взята в рамку, показывающую ее размеры. После масштабирования она повернута на 90° .

3.2. Верстка таблиц

L^AT_EX использует два метода создания таблиц.

В первом задается шаблон строки с ячейками фиксированной ширины, по которому затем формируются все строки. Таким образом верстается таблица с фиксированной шириной колонок, строки которой между собой не связаны, поэтому ее части могут располагаться на разных страницах. Этот метод реализован в окружении `tabbing`, описанном в книгах [2–4, 8].


Во втором методе шаблон строки задает только метод выравнивания ячеек, размеры которых изменяются в соответствии с размерами помещенного в них текста. Ширину колонки определяет ее самая широкая ячейка. В большинстве случаев она не фиксируется и изменяется по мере заполнения таблицы. Сверстанная таблица становится боксом, встраиваемым в текстовую строку по общим правилам.

Второй метод, обладающий существенно большей гибкостью и универсальностью, реализован с помощью окружений `array` и `tabular`, имеющих одинаковый синтаксис. Он используется чаще всего, поэтому рассмотрим его подробно, но сначала, на

представленном ниже примере окружения `tabular`, сформулируем основные правила верстки таблиц.

Колонки с левым, правым выравниванием и центрированием обозначаются идентификаторами `l`, `r` и `c`. Список идентификаторов помещается в аргумент окружения `tabular`, который определяет число колонок, метод их выравнивания и порядок следования. В приведенном примере формируется таблица с тремя колонками, выровненными слева, по центру и справа:

```
begin{tabular}{|l|c|r}
\hline
\tiny 2 & 3 & \\
10 & & 1000 \\
\hline
& & \\
\hline
$y_i^2$ & \tiger & \fbox{a} \\
\hline
& \bf ab & cd \\
\hline
\end{tabular}
```

2	3	
10	100	1000
y_i^2		a
	ab	cd

Ячейки разделяются символом `&`, а строки командой `\\`. В конце строки пустые ячейки можно опустить, как это сделано в третьей строке примера, поставив команду `\\` после последней заполненной ячейки, однако в начале строки пустые ячейки указываются всегда, так как это необходимо для правильного подсчета колонок. Количество строк в таблице не регламентируется, их можно свободно добавлять и удалять.

Ячейки должны содержать часть текстовой строки, в которой могут находиться строчные формулы, боксы, иллюстрации, как, например, в третьей строке таблицы, а также министраницы и даже другие таблицы. Каждая ячейка составляет группу, поэтому изменения шрифтов и других параметров, сделанные в одной из них, в другие не передаются. Это показывают соседние ячейки в первой и последней строках таблицы.

Строки и колонки можно отчеркивать. Для этого в аргументе окружения `tabular` используется символ-разделитель `|`, рисующий вертикальную линию сверху донизу таблицы, а команда

`\hline` проводит горизонтальную линию на всю ее ширину. При отчеркивании сверху команды `\hline` должны стоять в начале строки, а при отчеркивании снизу — после команды `\`. Количество отчеркивающих линий соответствует числу использованных команд `\hline` или разделителей `|`. Линии не накладываются друг на друга, а рисуются с небольшим зазором. Команда

```
\cline{диапазон колонок}
```

отчеркивает часть колонок. Она должна стоять после перевода строки. В ее аргументе указываются номера колонок, разделенные дефисом, (`\cline{1-2}` в примере), а если отчеркивается одна колонка, ее номер указывается дважды (`\cline{1-1}` и `\cline{3-3}` в примере). Чтобы отчеркнуть несколько диапазонов колонок, нужно использовать несколько команд `\cline`, перечисляемых в произвольном порядке. Линии `\cline`, отчеркивающие одну и ту же колонку, накладываются друг на друга, а также и на линию `\hline`, поэтому команды `\hline` и `\cline` не стоит использовать совместно.

3.2.1. Выравнивание и разделение колонок

Рассмотрим синтаксис окружения `tabular` в деталях:

```
\begin{tabular}[привязка]{шаблон строки}
  ячейка 1 & ячейка 2 & ...\\
  .....
\end{tabular}
```

Параметр `привязка` определяет встраивание таблицы в текстовую строку. Он может принимать следующие значения:

- b** — низ таблицы выравнивается с базисной линией;
- c** — таблица центрируется относительно базисной линии;
- t** — верх таблицы выравнивается с базисной линией.

По умолчанию бокс с таблицей центрируется в строке.

При использовании параметров **b** или **t** линии, рисуемые командами `\hline` и `\cline{•}`, влияют на привязку таблиц. Яв-

ляясь строками таблиц, они выравниваются с базисной линией. Без отчеркивания в текстовую строку встраивается первая или последняя строка таблицы. Это демонстрируют следующие примеры:

```

      b b
      b b
текст .....
      c c
      c c
      t t
      t t
      b b
..... текст b b текст c c текст t t текст .....
      c c
      t t
  
```

В аргументе окружения `tabular` указывается список идентификаторов колонок и разделителей, между которыми можно ставить пробелы, не влияющие на форматирование таблицы:

Идентификаторы колонок	Разделители
<code>c</code> — центрирование	<code> </code> — вертикальная линия
<code>l</code> — левое выравнивание	<code>@{текст}</code> — текст
<code>r</code> — правое выравнивание	
<code>p{ширина}</code> — заданная ширина	

По умолчанию колонки разделяются небольшим пустым полем, в котором разделитель `|` рисует вертикальную линию. Разделитель `@{текст}` заменяет пустое поле заданным текстом:

```

\begin{tabular}{|r @{*} l|} \hline
1 & 2 & \\
30 & 40 & \\
500 & 600 & \\ \hline
\end{tabular}
  
```

1*2
30*40
500*600

Как уже говорилось, ячейки колонок `c`, `l` и `r` содержат только одну текстовую строку. Ячейки колонок `p{ширина}` имеют фиксированную ширину и могут содержать несколько строк. Помещенный в них текст автоматически нарезается на строки заданной длины и выравнивается с обеих сторон. Для разрыва строк

можно пользоваться командой `\linebreak`. В строку таблицы встраивается первая строка ячеек `p{•}`:

<pre>\begin{tabular}{lp{8em}} \hline 1 & Короткая строка.\\ 2 & Более длинная строка.\\ 3 & Текст ячейки выравнивается...\\ \hline \end{tabular}</pre>	<table style="width: 100%; border-collapse: collapse;"> <tr><td style="padding: 2px 10px 2px 0;">1</td><td style="padding: 2px 0 2px 10px;">Короткая строка.</td></tr> <tr><td style="padding: 2px 10px 2px 0;">2</td><td style="padding: 2px 0 2px 10px;">Более длинная строка.</td></tr> <tr><td style="padding: 2px 10px 2px 0;">3</td><td style="padding: 2px 0 2px 10px;">Текст ячейки выравнивается и разбивается на несколько строк.</td></tr> </table>	1	Короткая строка.	2	Более длинная строка.	3	Текст ячейки выравнивается и разбивается на несколько строк.
1	Короткая строка.						
2	Более длинная строка.						
3	Текст ячейки выравнивается и разбивается на несколько строк.						

3.2.2. Выравнивание ячеек

Колонки `s`, `l` и `r` выравниваются пружинами `\hfil`, неявно стоящими в каждой ячейке. В колонках `l` и `r` они, соответственно, стоят справа и слева, а центрированную колонку сжимают с обеих сторон. Используя дополнительные пружины, можно регулировать выравнивание отдельных ячеек, не затрагивая при этом другие:

<pre>\begin{tabular}{ l c r } \hline 1 & 2 & 3 \\ 1111 & 2222 & 3333 \\ \hfil 1 & \hfil 2 & 3\hfil\mbox{} \\ \hfill 1 & \hfill 2 & 3\hfill\mbox{} \\ \hline \end{tabular}</pre>	<table border="1" style="border-collapse: collapse; text-align: center; width: 150px; height: 100px;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>1111</td><td>2222</td><td>3333</td></tr> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>1</td><td>2</td><td>3</td></tr> </table>	1	2	3	1111	2222	3333	1	2	3	1	2	3
1	2	3											
1111	2222	3333											
1	2	3											
1	2	3											

Разберем приведенный выше пример. Первая и вторая строка таблицы иллюстрируют методы выравнивания колонок. В ячейки третьей строки мы поставили пружины `\hfil`, при этом в крайних колонках цифры, оказавшись между одинаковыми пружинами, сместились в центр. Во второй колонке двойка смещена вправо на $2/3$ пустого поля, так как слева она зажата двумя одинаковыми пружинами, а справа — одной. Пример показывает, что только левая граница колонки служит упором пружин, справа же он отсутствует, поэтому в последней колонке в качестве упора

использован пустой бокс. В ячейках четвертой строки используются жесткие пружины `\hfill`, смявшие «неявные» пружины и отжавшие цифры к границам колонок.

3.2.3. Объединение колонок

Ячейки можно объединять как по горизонтали, так и по вертикали. Для объединения ячеек, находящихся в общей строке, имеется команда

```
\multicolumn{n}{описание ячейки}{текст}
```

с тремя аргументами. В первом указывается число объединяемых ячеек. Во втором — разделители, окружающие объединенную ячейку, и метод ее выравнивания, задаваемый одним из идентификаторов. В третий аргумент помещается текст ячейки.

Приведем пример использования команды `\multicolumn` для формирования заголовка таблицы:

```
\begin{tabular}{lccr@{ = }l} & & & & \toprule
Физические & \multicolumn{2}{c}{Единицы} & & \\
& \multicolumn{2}{c}{Соотношение} & \\\ \cmidrule{2-3}
величины & СИ & СГС & & \\
& \multicolumn{2}{c}{единиц} & \\\ \midrule
Длина & м & см & 1 м & 100 см & \\\
Масса & кг & г & 1 кг & 1000 г & \\\
Сила & Н & дина & 1 Н & $10^5$ дин & \\\
Энергия & Дж & эрг & 1 Дж & $10^7$ эрг & \\\ \bottomrule
\end{tabular}
```

Физические величины	Единицы		Соотношение единиц
	СИ	СГС	
Длина	м	см	1 м = 100 см
Масса	кг	г	1 кг = 1000 г
Сила	Н	дина	1 Н = 10 ⁵ дин
Энергия	Дж	эрг	1 Дж = 10 ⁷ эрг

В первой строке заголовком «Единицы» мы объединили вторую и третью колонки, содержащие обозначения двух систем единиц физических величин. Заголовок «Соотношение единиц» сделан путем объединения четвертой и пятой колонок в первой и второй строке. Между четвертой и пятой колонками с помощью разделителя `@{ = }` вставлен знак равенства, а сами колонки прижимаются к нему слева и справа.

В соответствии с правилами оформления таблиц в журнальных публикациях в приведенном примере отсутствуют вертикальные отчеркивающие линии. Для отделения заголовка и отчеркивания таблицы сверху и снизу использованы команды:

`\midrule[толщина]` ,

`\toprule[толщина]`

и

`\bottomrule[толщина]` ,

вводимые пакетом `booktabs` [14]. Они заменяют стандартную команду `\hline`, во многих случаях давая лучший результат. Их параметр позволяет регулировать толщину линий.

Для отчеркивания ячеек данный пакет определяет команду

`\cmidrule[толщина](ширина){диапазон колонок}` ,

позволяющую регулировать не только толщину линии, но и ее ширину. Отметим, что второй параметр этой команды заключен в нестандартные скобки.

3.2.4. Объединение строк

Непосредственно в ядре L^AT_EX отсутствуют средства объединения строк таблицы. Такую возможность предоставляет пакет `multirow` , который вводит с этой целью команду `\multirow`. Объединенная ячейка может содержать несколько строк текста. Покажем это на примере:

1	2	3	4
2	a	b	d
3		c	e
4			f

```

\begin{tabular}{|l|l|p{.5em}|l|}
1 & 2 & & 3 & & 4 \\ \hline
2 & \multirow{3}*{a} & \multirow{3}={b\ c} & & & \\ \cline{1-1}
3 & & & & & \\ \cline{1-1}
4 & & & \multirow{-3}{.5em}{d\ e\ f} & & \\ \hline
\end{tabular}

```

В первом столбце и строке таблицы мы пронумеровали ячейки, а в остальных создали ячейки, объединяющие по три строки каждой колонки. Текст ячейки второго столбца состоит из одной строки, третьего — из двух, а четвертого — из трех.

Разберем теперь синтаксис и действие команды

```
\multirow[привязка]{n}[распорки]{ширина}[смещение]{текст}
```

Ее первый параметр определяет метод вертикального выравнивания объединенной ячейки, который задается идентификаторами:

- c — текст центрируется по вертикали (значение по умолчанию);
- b — последняя строка текста выравнивается с нижней строкой таблицы;
- t — первая строка текста встраивается в верхнюю строку таблицы.

Параметр `распорки`, используемый редко, предназначен для настройки высоты объединяемых строк, имеющей сложный механизм с замысловатой логикой, описанной в документации пакета `multirow` [15].

Параметр `смещение` служит для подстройки вертикального выравнивания. В нем задается высота, на которую нужно сместить текст ячейки. При положительной длине он смещается вверх, а при отрицательной — вниз.

В первом аргументе команды `\multirow` указывается количество объединяемых строк, причем задающее его число (как показано в примере) может быть отрицательно. При положительном числе команда `\multirow` должна стоять в первой объединяемой строке, а при отрицательном — в последней. В любом случае остальные ячейки объединяемых строк должны быть пусты.

Второй аргумент регулирует ширину объединенной ячейки. Ее можно задать явно, указав в аргументе необходимую длину. Специальное значение аргумента `=` позволяет наследовать ширину колонки `r{•}`, как это сделано для третьей колонки. Еще одно специальное значение `*` предназначено для создания ячейки переменной ширины, которая может содержать только одну строку, как показано на примере второй колонки.

Декларация `\multirowsetup` хранит код, выполняемый перед форматированием ячейки `\multirow`, который по умолчанию состоит из декларации `\raggedright`, обеспечивающей левое выравнивание текста. Изменив данную команду, можно задать новые параметры верстки, например курсивное начертание центрированного текста:³

```
\renewcommand\multirowsetup{\centering\it}.
```

Новые параметры можно задать непосредственно перед командой `\multirow`, тогда изменения ограничатся только создаваемой ячейкой.

На этом закончим обсуждение основных ресурсов верстки таблиц. Дополнительные возможности описаны в гл. 8.

3.3. Плавающие объекты

Внимательный читатель возможно обратил внимание, что в данной книге рисунки и таблицы находятся рядом со ссылкой на них, что соответствует правилам российского книгоиздания. Однако такое расположение не типично для L^AT_EX.

Рисунки или таблицы обычно имеют довольно большие размеры и потому их помещают в «плавающие» объекты, положение которых в документе может отличаться от порядка следования в программе. Компилятор пытается оптимально распределить плавающие объекты, размещая их обычно вверху страниц поблизости от того места, которое они занимают в программе, так

³ Настройка команд описана в разд. 6.1.



Рис. 3.2. Стандартная компоновка рисунка:

```
\begin{figure}  
  \centering  
  \includegraphics{tiger}  
  \caption{Стандартная компоновка рисунка:}  
  \label{f:Figure}  
\end{figure}
```

как там они требуют минимум места.⁴ Если же их количество велико, а объем текста мал, они сводятся все вместе в конец документа. Использование в рукописи статьи средств, управляющих размещением плавающих объектов, не приветствуется. При необходимости их расставит по местам технический редактор.

На примере рис. 3.2 покажем автоматическое размещение плавающего объекта по правилам ЛАТ_ЭX. В тексте программы он находится непосредственно после этого абзаца.

Из примера видно, что помимо самой иллюстрации плавающий объект может содержать подпись, метку и любой дополнительный текст, который в примере представлен кодом, генерирующим рисунок.

Плавающие объекты создают окружения, предназначенные для размещения таблиц и рисунков:

```
\begin{table}[положение] , \begin{table*}[положение] ,  
\begin{figure}[положение] , \begin{figure*}[положение] .
```

⁴ Рисунки и таблицы отбиваются от окружающего текста вертикальными пробелами, которые в начале и конце страницы игнорируются, поэтому они прижимаются к краю страницы, экономя место для текста.

При многоколоночном наборе рисунки и таблицы, помещенные в окружения `figure*` или `table*`, занимают всю ширину страницы, а в `figure` или `table` — ширину колонки.

Положение плавающего объекта можно подстроить, используя следующие идентификаторы:

- `t` — поместить вверху колонки или страницы;
- `b` — поместить внизу колонки или страницы;
- `h` — разместить в соответствии с порядком следования в тексте;
- `p` — вынести на специальную страницу, содержащую плавающие объекты.

В параметре можно задать сразу несколько идентификаторов, например `\begin{figure}[tbh]`, среди которых компилятор выберет оптимальный.

Распределение плавающих объектов по страницам сильно влияет на качество верстки документа в целом, поэтому их размещение не всегда следует заданному параметру. Модификатор `!` усиливает его действие. Например, таблица, помещенная в окружение `\begin{table}[h!]`, должна появиться среди текста, окружающего ее в программе. В большинстве случаев модификатор `!` позволяет добиться желаемого результата, но если сверстанная страница окажется слишком пустой или переполненной, параметр все же может игнорироваться.

Рисунок или таблицу можно подписать с помощью команды

```
\caption[краткая подпись]{подпись} .
```

Ее использование вне окружений `figure` или `tabular` вызовет ошибку компиляции, так как она определена только внутри них. Аргумент команды содержит подпись, а в параметре можно указать ее сокращенный вариант, используемый в списке рисунков или таблиц. В конце подписи в документах, написанных по-русски, точка не ставится, а в англоязычных она, как правило, ставится.

Параметр позволяет убрать из подписи элементы, которые не должны отражаться в списках. Например, табл. 2.6 имеет

примечание, а ее подпись содержит команду `\footnotemark`, генерирующую номер примечания. В то же время в списке таблиц нет ссылки на примечание, так как в варианте подписи, помещенном в параметр команды `\caption`, команда `\footnotemark` отсутствует.

С командами `\caption` ассоциированы счетчики `figure` и `table`. Именно подпись является объектом ссылки на рисунок или таблицу, поэтому непосредственно после команды `\caption` следует поставить метку. Рекомендуем снабдить ее префиксом, например `\label{f:•}` или `\label{t:•}`. При формировании ссылок в программе `TeXStudio` всплывает отсортированный список меток, в котором одинаковые префиксы группируются вместе. Это значительно облегчает поиск нужной метки.

Плавающий объект компоуется как обычный текст, в котором рисунки и таблицы представляют собой боксы, находящиеся в текстовой строке, а подпись является отдельным абзацем. Положение подписи относительно рисунка (или таблицы) задает порядок следования команд `\caption` и `\includegraphics` (или окружения `tabular`). Если команда `\caption` стоит впереди, подпись выводится сверху, а если позади, то снизу, как показано ранее.

Чтобы вывести подпись сбоку от рисунка, их нужно поместить в министраницы. Разберем конструкцию плавающего объекта, обеспечивающего такую компоновку рисунка 3.3.

Рис. 3.3. Компоновка рисунка и подписи, расположенной слева:



```
\begin{figure}
  \begin{minipage}{.55\linewidth}
    \caption{Пример компоновки рисунка и подписи,
             расположенной слева:}
    \label{f:CaptionAtTheRight}
  \end{minipage}
```

```

\hfill
\begin{minipage}{.43\linewidth}
  \centering
  \includegraphics[width=.45\linewidth]{tiger}
\end{minipage}
\end{figure}

```

Ее основу составляют две министраницы, стоящие в одной строке, а чтобы они уместились в ней, их суммарная ширина сделана немного меньше ее длины. В первую министраницу помещена подпись, а во второй находится картинка. Между ними поставлена жесткая пружина `\hfill`, отжимающая подпись влево.

Ширина левой министраницы регулирует ширину подписи, а правой — размеры картинка, которая масштабируется до 45% ее ширины.⁵ Декларация `\centering` делает симметричные поля вокруг картинка.

Так как министраницы автоматически центрируются относительно базисной линии строки, подпись оказывается центрированной по высоте рисунка.

Усложним компоновку, добавив еще одну картинку (рис. 3.4), и чтобы предать композиции законченность, отразим ее с помощью команды

```
\reflectbox{текст}.
```

Ее аргументом может быть текст длиной не более одной строки.



Рис. 3.4. Компоновка рисунка, состоящего из двух картинок и подписи, расположенной справа от них

Как показано в приведенном внизу коде, чтобы уместить еще одну картинку, мы увеличили ширину первой министраницы,

⁵ Длина строки равна ширине министраницы.

уменьшив вторую. Чтобы сделать одинаковыми поля вокруг картинок, вновь использована декларация `\centering`, а между ними вставлена пружина `\hfil`.

```
\begin{figure}
  \begin{minipage}{.58\linewidth}
    \centering
\reflectbox{\includegraphics[width=.4\linewidth]{tiger}}
    \hfil
      \includegraphics[width=.4\linewidth]{tiger}
  \end{minipage}
\hfill
  \begin{minipage}{.4\linewidth}
    \caption{Пример компоновки рисунка, состоящего
      из двух картинок и подписи, расположенной
      справа от них}
    \label{f:Mirror}
  \end{minipage}
\end{figure}
```

Глава 4

Верстка формул

Высокое качество верстки формул — одно из главных достоинств \TeX . Алгоритмы обработки текста и математических выражений сильно различаются. Различаются также наборы шрифтов и команд, которыми оперирует компилятор, поэтому математические символы нельзя использовать в тексте, а русские буквы — в формулах. Большинство команд, используемых для форматирования текста, неприменимо в формулах, и наоборот.

Основными элементами формул являются переменные, функции, операторы, скобки, знаки различных соотношений, индексы, корни, дроби и т. д. Каждый элемент имеет свое шрифтовое оформление, размер и отбивку, которые очень хорошо сбалансированы с окружающими элементами. Символы и основные конструкции задаются командами, а сложные конструкции типа матриц, систем уравнений и т. п. — окружениями. Пробелы также регулируются командами, а пробелы текста программы полностью игнорируются. Как правило, в формулах имеются команды, аргументы которых содержат другие команды со своими аргументами, и «уровень вложенности» таких команд может быть довольно велик. Например, на «третьем уровне» находятся индексы переменной, стоящей в числителе или знаменателе дроби, являющейся подкорненным выражением. Поскольку аргументы

закрываются в фигурные скобки, возникает много вложенных скобок, а потеря хотя бы одной из них ведет к ошибке компиляции. Поэтому рекомендуем по возможности избегать лишних скобок и, используя алгоритм автоматического определения аргумента команд, набирать x_i^2 вместо $x_{\{i\}^{\{2\}}$, или $\sqrt{2}$ вместо $\sqrt{\{2\}}$. Это намного упростит текст программы и уменьшит вероятность ошибок. Набирая скобки, придерживайтесь золотого правила: водите сразу пару — открывающую и закрывающую.

Набор доступных команд и окружений определяется списком загружаемых пакетов. К числу стандартных ресурсов принадлежит коллекция пакетов `amslatex` Американского математического общества. Описываемые далее средства верстки формул подразумевают использование пакетов `amssymb` и `amsmath` из этой коллекции. Первый из них автоматически загружает пакет `amsfonts` и существенно расширяет набор символов, а второй определяет большое число дополнительных конструкций. Мы не будем разделять ресурсы стандартного L^AT_EX и `amslatex`, поэтому подчеркнем еще раз, что преамбула документа должна содержать команду

```
\usepackage{amssymb,amsmath}.
```

Простые формулы, находящиеся в текстовой строке, заключаются в символы `$...$`, например

$$\sqrt{x^2} = \pm x \rightsquigarrow \sqrt{x^2} = \pm x.$$

Будем называть их *строчными формулами*. Вместо символа `$` формулу можно заключить в команды `\(...\)`, но они менее удобны и потому практически не используются.

Громоздкие формулы выносятся из строки в отдельный абзац, и потому будут называться *вынесенными*. Приведем пример такой формулы:

$$\int_0^1 \frac{x^2 dx}{\sqrt{1-x^2}} = \frac{1}{2} \left(\arcsin x - x \sqrt{1-x^2} \right) \Big|_0^1 = \frac{\pi}{4}.$$

Вынесенные формулы бывают *нумерованные* и *ненумерованные*. Нумерованные формулы создаются с помощью окружений, обсуждаемых в разд. 4.3. Ненумерованные формулы заключаются

либо в двойные символы `$$...$$`, либо между команд `\[...]`. Конструкция `$$...$$` наследована из $\text{T}_{\text{E}}\text{X}$, а `\[...]` определена в $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$. Рекомендуем пользоваться последней из них, так как в некоторых случаях она обрабатывается более корректно. Например, при использовании параметра `fleqn` в стандартных классах документов (с. 24) формулы `\[...]` прижмутся к левому полю, а формулы `$$...$$` останутся центрированными.

Независимо от того, как создана формула, она является частью фразы и должна выделяться знаками пунктуации. В конце формулы, завершающей предложение, ставится точка.

Чтобы не загромождать текст, в представленных далее примерах формул опущены символы `$` или `\[` и `\]`. Приведены лишь команды, необходимые для понимания сути примеров.

4.1. Основные элементы и конструкции

4.1.1. Буквы, символы и векторы

В формулах определены латинские буквы, цифры и основные символы `() [] + - = ! ? | / < > * ; ' ,`, вводимые непосредственно с клавиатуры. Большая часть математических символов, в том числе фигурные скобки и греческие буквы, задана командами. Чтобы не загромождать текст, в данном разделе приведены только сами символы, а таблицы, содержащие их шестнадцатеричные коды и соответствующие им команды, вынесены в прил. В. Современные графические оболочки содержат таблицы символов, подобные приведенным далее. В `TexStudio` они находятся в многофункциональной панели, находящейся в левой части окна редактора. Чтобы вывести их, нажмите расположенную слева иконку со звездочкой. Меню, расположенное справа от строки поиска, позволяет переключаться между различными таблицами.

В стандартном математическом шрифте цифры, скобки и другие символы имеют прямое начертание, а буквы — курсивное. Дополнительные шрифты, называемые *математическими алфавитами*, обеспечивают широкий выбор обозначений переменных.

Команды и символы математических алфавитов

Стандартный математический шрифт

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxy
 0123456789

Жирный курсив `\boldsymbol{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxy
0123456789

Узкий курсив `\mathit{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxy
 0123456789

Жирный узкий курсив `\boldsymbol{\mathit{•}}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxy
0123456789

Прямой шрифт `\mathrm{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
 abcdefghijklmnopqrstuvwxy
 012345678

Жирный прямой шрифт `\mathbf{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxy
0123456789

Шрифт Serif `\mathsf{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
 abcdefghijklmnopqrstuvwxy
 0123456789

Наведите курсор на символ, чтобы открыть подсказку.

Жирный Serif `\boldsymbol{\mathsf{•}}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

Шрифт Typewriter `\mathtt{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

Готический шрифт `\mathfrak{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

Жирный готический шрифт `\boldsymbol{\mathfrak{•}}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

Рукописный шрифт¹ `\mathscr{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Жирный рукописный шрифт¹ `\boldsymbol{\mathscr{•}}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Каллиграфический шрифт `\mathcal{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Жирный каллиграфический шрифт `\boldsymbol{\mathcal{•}}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Ажурный шрифт `\mathbb{•}`

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Наведите курсор на символ, чтобы открыть подсказку.

¹ Чтобы использовать рукописный шрифт, загрузите пакет `eucal` с параметром `mathscr`. Загрузка пакета без параметра заменит каллиграфический шрифт рукописным.

Буквы математических алфавитов вводятся с помощью команд, приведенных в табл. 4.1 и прил. В.1, например

$$\backslash\mathfrak{AaBb} \rightsquigarrow \mathfrak{AaBb}.$$

Символы, заданные командами, как правило, остаются неизменными при смене шрифта,

$$\backslash\mathfrak{x\le\alpha D} \rightsquigarrow \mathfrak{x \le \alpha D},$$

тем не менее чтобы избежать конфуза, данную операцию стоит применять лишь к отдельным буквам, следя за корректностью ее выполнения. В шрифтах `\mathcal`, `\mathscr` и `\mathbb` есть только заглавные буквы, а на месте строчных букв и цифр стоят совершенно другие символы, поэтому их использование может привести к забавным результатам: `\mathbb{A_i}` $\rightsquigarrow \mathbb{A}_i$, или `\mathcal{x^2}` $\rightsquigarrow \mathcal{x}^\epsilon$.

Имеются различные варианты греческих букв, собранные в табл. 4.2 и прил. В.10

Таблица 4.2

Греческие буквы

Г	Δ	Θ	Λ	Ξ	Π	Σ	Υ	Φ	Ψ	Ω	Г	Δ	Θ	Λ	Ξ	Π	Σ	Υ	Φ	Ψ	Ω	Γ				
α	β	γ	δ	ε	ε	ζ	η	θ	ι	κ	κ	λ	μ	ν	ξ	π	ρ	ρ	σ	σ	τ	υ	φ	χ	ψ	ω
Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	Φ	Ψ	Ω	Г	Δ	Θ	Λ	Ξ	Π	Σ	Υ	Φ	Ψ	Ω					
α	β	γ	δ	ε	ε	ζ	η	θ	ι	κ	κ	λ	μ	ν	ξ	π	ρ	ρ	σ	σ	τ	υ	φ	χ	ψ	ω

Наведите курсор на символ, чтобы открыть подсказку.

Переменные можно снабдить акцентами, представленными в табл. 4.3, например `f_x = m\dot{v}_x` $\rightsquigarrow f_x = m\dot{v}_x$.

Таблица 4.3

Математические акценты

â	ă	à	á	ã	ā	ǎ	ǎ	á	ä	ä	¨	¨	abc	abc
---	---	---	---	---	---	---	---	---	---	---	---	---	-----	-----

Наведите курсор на символ, чтобы открыть подсказку.

Хотя среди акцентов имеется и значок вектора:

$$\backslash\vec{\alpha} \rightsquigarrow \vec{\alpha},$$

его следует использовать только в каких-то специальных случаях, так как стандартом обозначения векторов является жирный шрифт. Для латинских букв это прямой шрифт `\mathbf{•}`:

$$\begin{aligned} \mathbf{a} &= a_x \mathbf{e}_x + a_y \mathbf{e}_y + a_z \mathbf{e}_z \end{aligned}$$

Векторные переменные, обозначенные буквами математических алфавитов или греческими буквами, вводят с помощью команды

`\boldsymbol{выражение}`,

как в данных примерах:

$$\begin{aligned} \boldsymbol{\xi}_x &= \boldsymbol{\xi} \mathbf{e}_x, \\ \boldsymbol{\mathfrak{H}} &= \mathbf{H}_a + \mathbf{H}_d. \end{aligned}$$

Команда `\boldsymbol` делает жирным не отдельный символ, а все выражение. Результат ее действия зависит от имеющихся шрифтов. Насыщенность символов не изменится, если в шрифтах отсутствует жирный вариант, например

$$\boldsymbol{\mathbb{B}} \rightsquigarrow \mathbb{B}.$$

Латинские буквы, производимые командами

$$\mathbf{abc} \rightsquigarrow \mathbf{abc} \text{ и } \boldsymbol{abc} \rightsquigarrow \mathbf{abc},$$

имеют разное начертание, поэтому для набора векторов не следует применять упрощенные конструкции типа:

$$\begin{aligned} \boldsymbol{\xi}_x &= \boldsymbol{\xi} \mathbf{e}_x, \\ \boldsymbol{\mathfrak{H}} &= \mathbf{H}_a + \mathbf{H}_d. \end{aligned}$$

В табл. 4.4 и 4.5 (см. также табл. В.13–В.15) собраны различные символы и бинарные операторы. Среди них наиболее часто используются постоянная Планка (\hbar `\hslash` или \hbar `\hbar`), символ бесконечности (∞ `\infty`), оператор градиента (∇ `\nabla`), дифференциал частной производной (∂ `\partial`), а также знаки умножения (\cdot `\cdot` и \times `\times`).

Таблица 4.4

Различные символы

∂	\hbar	\hbar	\Im	\Re	\wp	\angle	\sphericalangle	\triangleleft	\square	\blacksquare	\diamond	\blacklozenge	\lozenge	\star
∞	ι	j	l	\mathbb{k}	\mathbb{U}	\diagdown	\diagup	\triangle	∇	\blacktriangle	\blacktriangledown	\spadesuit	\heartsuit	\clubsuit
∇	\aleph	\beth	\beth	\beth	\eth	\backslash	$/$	\exists	\notin	\emptyset	\emptyset	\checkmark	\checkmark	\spadesuit
	$\text{\textcircled{R}}$	$\text{\textcircled{S}}$	\textcent	\textpound	\textcyrillic	\perp	\top	\mathbb{C}	\neg	b	$\#$	\dagger	\sphericalangle	

Наведите курсор на символ, чтобы открыть подсказку.

Таблица 4.5

Бинарные операторы

\pm	\mp	\lessgtr	\gtrless	Υ	\rceil	\lrcorner	\ulcorner	$\bar{\wedge}$	$\bar{\vee}$	$\bar{\cap}$	\boxplus	\boxtimes
\times	\sphericalangle	\triangleleft	\triangleright	\cap	\cup	\sqcap	\sqcup	\cap	\cup	\oplus	\otimes	\odot
\times	\times	\triangleleft	\triangleright	\oplus	\ominus	\otimes	\odot	\ominus	\otimes	\oslash	\odot	\bigcirc
\dagger	\ddagger	\triangleleft	\triangleright	\circ	\bullet	\diamond	\diamond	\cdot	$*$	\diagdown	∇	\triangle
		\times	\cdot	\div	$*$	\dagger	\sphericalangle	\top	\star	\diagdown	Π	

Наведите курсор на символ, чтобы открыть подсказку.

Таблица 4.6

Соотношения

\equiv	\approx	$>$	$<$	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
\circ	\approx	$>$	$<$	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx
$\#$	\approx	\gg	\ll	\gg	\ll	\cup	\cap	γ	γ	\asymp	\approx	\approx

Наведите курсор на символ, чтобы открыть подсказку.

Таблица 4.7

Стрелки (соотношения)

\leftarrow	\rightarrow	\leftrightarrow	\longleftarrow	\longrightarrow	\longleftarrow	\longrightarrow	\Leftrightarrow	\Leftarrow	\Rightarrow
\nleftarrow	\nrightarrow	\nleftrightarrow	\nleftarrow	\nrightarrow	\nleftarrow	\nrightarrow	\nLeftrightarrow	\nLeftarrow	\nRightarrow
$\overleftarrow{\quad}$	$\overrightarrow{\quad}$	$\overleftrightarrow{\quad}$	$\overleftarrow{\quad}$	$\overrightarrow{\quad}$	$\overleftarrow{\quad}$	$\overrightarrow{\quad}$	$\overleftrightarrow{\quad}$	$\overleftarrow{\quad}$	$\overrightarrow{\quad}$
\uparrow	\downarrow	\updownarrow	\dashleftarrow	\dashrightarrow	\uparrow	\uparrow	\updownarrow	\Uparrow	\Downarrow
\Leftarrow	\Rightarrow	\mapsto	\P	\Q	\downarrow	\downarrow	\rightsquigarrow	\Lleftarrow	\Rrightarrow
\Leftrightarrow	\Rrightarrow	\mapsto	\circlearrowleft	\circlearrowright	\uparrow	\uparrow	\rightsquigarrow	\nearrow	\searrow
\Uparrow	\Downarrow		\curvearrowleft	\curvearrowright	\rightsquigarrow	\circ		\swarrow	\nwarrow

Наведите курсор на символ, чтобы открыть подсказку.

Табл. 4.6, 4.7 (см. также табл. В.16–В.17) содержат символы различных соотношений, к которым относится и большой набор различных стрелок.

Наиболее часто используются правая стрелка (\rightarrow `\to`), знаки пропорциональности (\propto `\propto`), тождества (\equiv `\equiv`), примерно (\simeq `\simeq`), порядка (\sim `\sim`), не равно (\neq `\neq`), а также различные варианты сравнения (\leq `\leq` и \geq `\geq`, \lesssim `\lesssim` и \gtrsim `\gtrsim`, \ll `\ll` и \gg `\gg`).

В табл. 4.8 и В.19 собраны различные варианты точек и многоточий.

Таблица 4.8

Точки и многоточия

$x \cdot x$	$x \cdot x$	$x \cdots x$	$x \cdot x$	$x \dots x$	$x \dots x$	$x \dots x$	$, \dots x$
$x : x$	$x : x$	$x \ddots x$	$+ \cdots x$	$\times \cdots x$	$\int_b^a \cdots x$		

Наведите курсор на символ, чтобы открыть подсказку.

4.1.2. Функции и основные конструкции

Кроме переменных, формулы содержат функции и конструкции типа верхних и нижних индексов, корней, дробей и т. п.

Функции выделяются прямым шрифтом и небольшой отбивкой, величина которой меняется в зависимости от представления аргумента и символа, стоящего перед функцией.

$\cos x$	$\arccos x$	$\operatorname{ch} x$	$\operatorname{cosh} x$	$\operatorname{csc} x$	$\operatorname{cosec} x$
$\sin x$	$\arcsin x$	$\operatorname{sh} x$	$\operatorname{sinh} x$	$\operatorname{sec} x$	
$\operatorname{tg} x$	$\operatorname{arctg} x$	$\operatorname{th} x$	$\operatorname{tanh} x$	$\operatorname{tan} x$	$\operatorname{arctan} x$
$\operatorname{ctg} x$	$\operatorname{arcctg} x$	$\operatorname{cth} x$	$\operatorname{coth} x$	$\operatorname{cot} x$	
$\exp x$	$\ln x$	$\lg x$		$y(x)$	$y \pmod{x}$
$\operatorname{arg} x$	$\operatorname{deg} x$	$\operatorname{dim} x$	$\operatorname{hom} x$		$y \bmod x$
$\operatorname{ker} x$	$\operatorname{P} x$	$\operatorname{sup} x$	$\operatorname{D} x$		$y \bmod x$

Наведите курсор на символ, чтобы открыть подсказку.

В табл. 4.9 приведен набор имеющихся в \LaTeX функций. Несмотря на свою обширность он не в состоянии охватить все их разнообразие, поэтому авторам предоставлена возможность самим создавать недостающие. Для этого достаточно набрать имя функции прямым шрифтом `\mathrm`, сравните, например,

`x \mathrm{sin} x \rightsquigarrow x \sin x` и `x \sin x \rightsquigarrow x \sin x`.

Команда

`\operatorname{функция}`

создает функцию непосредственно при ее использовании, например `\operatorname{erf} x \rightsquigarrow \operatorname{erf} x`.

Часто используемую функцию можно задать в виде команды. Это позволяет сделать конструкция

`\DeclareMathOperator{команда}{функция}`,

определяющая новую команду для вывода функции. Определения новых функций лучше разместить в преамбуле. В следующем примере для функции `\G` вводится команда `\G`:

```
\DeclareMathOperator{\G}{grad}
\begin{document}
... x \G(x)/y ...
```

Корень n -й степени задается командой

`\sqrt[степень]{выражение}`,

например: $\sqrt[4]{xx^*} = \sqrt{\pm|x|} \rightsquigarrow \sqrt[4]{xx^*} = \sqrt{\pm|x|}$.
 Степень квадратного корня не указывается, а у подкоренного выражения, состоящего из одного символа, скобки можно опустить:

$$\sqrt{x} \rightsquigarrow \sqrt{\xi}.$$

Дробь задается командой

$$\frac{\text{числитель}}{\text{знаменатель}}.$$

Размер символов числителя и знаменателя зависит от типа формул. Чтобы строчные формулы не раздвигали строки, в дробях используется мелкий шрифт $\frac{1+x}{1+x} \rightsquigarrow \frac{1+x}{1+x}$. А дроби, находящиеся в числителе или знаменателе, уменьшаются еще сильнее:

$$x + \frac{x+1}{x + \frac{x+1}{x + \frac{1}{x + 1}}}$$

Команды $\frac{\bullet}{\bullet}$ и $\frac{\bullet}{\bullet}$ позволяют регулировать размер дробей. Предыдущий пример будет смотреться намного лучше при их использовании:

$$x + \frac{x+1}{x + \frac{x+1}{x + \frac{1}{x + 1}}},$$

$$x + \frac{x+1}{x + \frac{x+1}{x + \frac{1}{x + 1}}}.$$

Числитель и знаменатель дробей $\frac{\bullet}{\bullet}$ и $\frac{\bullet}{\bullet}$ имеют такой же размер, как у дроби $\frac{\bullet}{\bullet}$, находящейся, соответственно, в строчной или вынесенной формуле (о размерах символов см. разд. 4.1.5).

Верхние и нижние индексы задаются командами

$$\overset{\text{выражение}}{} \quad \text{и} \quad \underset{\text{выражение}}{},$$

например $x_i^2 \rightsquigarrow x_i^2$. Порядок следования индексов произволен, но при наборе следует различать индексы индексированного выражения:

$$x^2_{x_i}_j \rightsquigarrow x^{2^2}_{x_i}_j$$

и индексы индексов:

$$x^{2^2} x_{i_j} \rightsquigarrow x^{2^2} x_{ij}.$$

Это различие, в частности, не позволяет использовать двойные индексы одинакового типа: x_{i_j} или x^{2^2} . Размер индексов уменьшается только до второго «уровня», начиная с которого он остается неизменным. Положение индексов определяется высотой и глубиной индексируемого выражения. Сравните

$$|(x_i^2)^{\alpha_\beta}| \rightsquigarrow (x_i^2)_{\alpha_\beta}^\alpha$$

и

$$|(x_i^2)^{\alpha_\beta}| \rightsquigarrow (x_i^2)_{\alpha_\beta}^\alpha.$$

В первом случае индексы α и β привязаны к закрывающей скобке, а во втором — к выражению в целом.

Команды индексов используются также для расстановки пределов у интегралов, сумм и т. д., например, $\int_a^b \rightsquigarrow \int_a^b$, $\sum_{i=1}^\infty \rightsquigarrow \sum_{i=1}^\infty$. В таких конструкциях иногда возникает необходимость в многострочных индексах. Их позволяет сделать команда:

$$\backslash\substack{\text{индекс1} \\ \text{индекс2} \dots}.$$

Строки индексов разделяются командой $\backslash\backslash$ и центрируются:

$$\sum_{\substack{0 < i < m, \\ \backslash\backslash 0 < j < n}} P(i, j) \rightsquigarrow \sum_{\substack{0 < i < m, \\ 0 < j < n}} P(i, j).$$

Окружение `subarray` позволяет настроить выравнивание многострочных индексов. Оно имеет аргумент, принимающий значение: c — центрирование, l или r — выравнивание слева или справа:

$$\sum_{\substack{\begin{subarray}{l} 0 < i < m, \\ \backslash\backslash 0 < j < n \end{subarray}}} P(i, j) \rightsquigarrow \sum_{\substack{0 < i < m, \\ 0 < j < n}} P(i, j).$$

4.1.3. Большие операторы и пределы

Отдельную группу объектов образуют большие операторы и функции, имеющие пределы. Особенности их поведения рассмотрим на примере интеграла, суммы и функции предела, а полный перечень этих объектов представлен в табл. 4.10 и В.21.

Большие операторы и функции, имеющие пределы

$\Sigma \Sigma$	$\cap \cap$	$\uplus \uplus$	$\odot \odot$	$\int \int$	$\iiint \iiint$
$\Pi \Pi$	$\cup \cup$	$\vee \vee$	$\oplus \oplus$	$\oint \oint$	$\iiiii \iiiii$
$\amalg \amalg$	$\sqcup \sqcup$	$\wedge \wedge$	$\otimes \otimes$	$\iint \iint$	$\int \dots \int \int \dots \int$
\lim	\limsup	$\overline{\lim}$	\liminf	$\underline{\lim}$	injlim
\det	\gcd	\inf	\max	\min	Pr
				$\underline{\lim}$	projlim
					$\underline{\lim}$

Наведите курсор на символ, чтобы открыть подсказку.

Прежде всего отметим, что операторы имеют разные размеры в вынесенных и строчных формулах:

$$\int \int \quad \text{и} \quad \Sigma \Sigma.$$

Обычно они снабжаются пределами, которые в вынесенных формулах могут располагаться либо сбоку, как у определенных интегралов, либо над и под ними, как у сумм и пределов:

$$\int_a^b \quad \sum_{i=0}^{\infty} \quad \lim_{x \rightarrow 0}$$

В строчных формулах, чтобы не раздвигать строки, пределы всегда проставляются сбоку: $\int_a^b \sum_{i=0}^{\infty} \lim_{x \rightarrow 0}$

С помощью команд `\limits` и `\nolimits` можно управлять положением индексов. Данные команды ставятся между оператором и его пределами. Использование `\limits` гарантирует расстановку пределов над и под оператором, а команда `\nolimits`, напротив, всегда размещает их сбоку:

$$\int_a^b \quad \sum_{i=0}^{\infty}$$

У некоторых операторов пределы могут находиться как слева, так и справа. Правильно расставить их помогает специальная конструкция:

`\sideset{левые индексы}{правые индексы}{оператор}`.

Ее аргументами являются команды расстановки индексов и большой оператор. Вот как выглядит пример ее использования:

$$\sideset{^1_2}{^3_4}\prod \rightsquigarrow \prod_2^1 \prod_4^3.$$

Независимо от типа формулы `\sideset` всегда выводит символ большого размера.

Для создания функций, имеющих пределы, определены команды

`\operatorname*{функция}`,

`\DeclareMathOperator*{команда}{функция}`.

4.1.4. Разделители

Особый тип объектов составляют разделители. Их набор приведен в табл. 4.11 и В.12.

Разделители

Таблица 4.11

()	[]	/	\			[]	⌈	⌋	↑	↓	↕
{	}	⟨	⟩					[]	⌊	⌋	↑	↓	↕

Наведите курсор на символ, чтобы открыть подсказку.

Так как разделители привычно ассоциируются со скобками, будем считать их синонимами. Наряду с обычными скобками очень часто используются угловые скобки:

`\langle x \rangle` \rightsquigarrow $\langle x \rangle$.

Размер скобок должен быть согласован с высотой заключенного в них выражения. Настроить его позволяет набор команд:

`\big` `\bigl` `\bigr` `\bigm`
`\Big` `\Bigl` `\Bigr` `\Bigm`
`\bigg` `\biggl` `\biggr` `\biggm`
`\Bigg` `\Biggl` `\Biggr` `\Biggm`

которые ставятся перед скобками и увеличивают их размер от `\big` до `\Bigg`. В примере показаны обычные скобки и все четыре варианта их увеличения:

$$\begin{array}{l} \backslash\Biggl(\backslashbiggl(\backslashBigl(\backslashbigl(\\ (t+y) \\ \backslashbigr)\backslashBigr)\backslashbiggr)\backslashBiggr) \end{array} \quad \left(\left(\left(\left(\left(t+y\right)\right)\right)\right)\right)$$

Команды с именами, заканчивающимися на `l`, `r` и `m`, обеспечивают дополнительную отбивку вокруг разделителя. Они предназначены, соответственно, для левых и правых скобок и символов, разделяющих две части выражения. Сравните, например:

$$\begin{array}{l} \frac{x}{y} \backslashbig/ \frac{y}{x} \rightsquigarrow \frac{x}{y} / \frac{y}{x} \\ \text{и} \\ \frac{x}{y} \backslashbigm/ \frac{y}{x} \rightsquigarrow \frac{x}{y} / \frac{y}{x}. \end{array}$$

С настройкой скобок вручную связано две проблемы. Во-первых, их максимальный размер ограничен, тогда как заключенные в них выражения могут оказаться очень громоздкими. Во-вторых, высота выражения может различаться в строчной и вынесенной формулах, и, если размер скобок зафиксирован, конструкция в целом может выглядеть в них по-разному. Например, дробь, охваченная скобками, выглядевшая в строчной формуле хорошо, `\bigl(\frac{b}{y}\bigr) \rightsquigarrow (\frac{b}{y})`, в вынесенной будет смотреться плохо: $(\frac{b}{y})$.

Обе проблемы устраняет специальная конструкция:

$$\backslashleft \bullet \text{ выражение } \backslashright \bullet .$$

Команды `\left` и `\right` ставятся перед левым и правым разделителем. Они служат метками границ выражения, высоту которого компилятор должен вычислить, чтобы поставить скобки нужного размера, который в данной конструкции не ограничен. Вот как будет выглядеть предыдущий пример при использовании конструкции `\left(\frac{b}{y}\right)` в строчной формуле $(\frac{b}{y})$ и в вынесенной формуле: $(\frac{b}{y})$.

Команды `\left` и `\right` используются только в паре, при этом они не могут находиться в разных строках или полях формул, имеющих сложную конструкцию, рассматриваемых в разд. 4.2 и 4.3. Разбиение пары командой перевода строки `\\` или знаком разделения полей `&` вызовет ошибку компиляции. В таком случае следует использовать две пары `\left` и `\right`, каждая из которых находится в своем поле, а отсутствующий разделитель заменить точкой.

В следующем примере длинное выражение, охваченное квадратными скобками, разбито на две строки. В каждой из них пары `\left` и `\right` содержат по одной скобке и одной точке. Открывающую скобку `\left[` «закрывает» команда `\right.` в конце первой строки, а начинающая вторую строку команда `\left.` «закрывается» скобкой `\right]`:

```
\begin{multline*}
\left[\frac{1}{2\operatorname{ch}(R/r)}
+\dots\right.
\left.+\dots
\frac{1}{2\operatorname{sh}(R/r)}\right],
\end{multline*}
```

$$\left[\frac{1}{2\operatorname{ch}(R/r)} + \dots + \dots \frac{1}{2\operatorname{sh}(R/r)} \right],$$

4.1.5. Размер символов в формулах

Формула является частью текста, поэтому размер ее символов согласуется с текущим размером текстового шрифта. Чтобы увеличить или уменьшить формулу нужно изменить размер шрифта перед ней, например, `{\footnotesize $x=y$}` $\rightsquigarrow x = y$ и `{\Large $x=y$}` $\rightsquigarrow x = y$. Это справедливо как для строчных, так и вынесенных формул.

В самих формулах определены всего три размера: `T` — для основных символов, `S` — для индексов и `SS` — для вторых индексов. В строчных формулах они примерно соответствуют текстовым размерам `\normalsize`, `\scriptsize` и `\tiny`.

Четыре комбинации расстановки размеров символов в дробях и индексах образуют математические стили.

`\displaystyle` — стиль вынесенных формул. Символы, стоящие в основной строке формулы и в дробях, имеют одинаковый размер Т. Индексы имеют размер S. Таким же размером набираются дроби, стоящие в числителе и знаменателе. Индексы индексов, дроби в индексах, а также индексы и дроби во «вложенных дробях» имеют размер SS:

$$2^{\frac{x}{2}}x^{2^x} + \frac{2^{\frac{x}{2}}x^{2^x}}{2^{\frac{x}{2}}x^{2^x} + \frac{2^{\frac{x}{2}}x^{2^x}}{2^{\frac{x}{2}}x^{2^x} + \frac{2^{\frac{x}{2}}x^{2^x}}{2+x}}} \quad \backslash\displaystyle.$$

`\textstyle` — стиль формул в строке. Символы основной строки формулы имеют размер Т. Индексы и дроби набираются размером S. Размер остальных символов равен SS:

$$2^{\frac{x}{2}}x^{2^x} + \frac{2^{\frac{x}{2}}x^{2^x}}{2^{\frac{x}{2}}x^{2^x} + \frac{2^{\frac{x}{2}}x^{2^x}}{2+x}} \quad \backslash\textstyle.$$

`\scriptstyle` — стиль индексов. Символы, стоящие в основной строке формулы, имеют размер S. Размер остальных символов равен SS:

$$2^{\frac{x}{2}}x^{2^x} + \frac{2^{\frac{x}{2}}x^{2^x}}{2+x} \quad \backslash\scriptstyle, \quad 2^{\frac{x}{2}}x^{2^x} \quad \backslash\scriptscriptstyle.$$

`\scriptscriptstyle` — стиль вторых индексов. Все символы имеют одинаковый размер SS.

При размере текстового шрифта `\tiny` в формулах остается только два размера символов. Более крупный используется в стилях `\displaystyle` и `\textstyle` для символов, стоящих в основной строке формулы, все же остальные символы, включая символы стилей `\scriptstyle` и `\scriptscriptstyle`, имеют более мелкий размер.

Перечисленные декларации позволяют использовать стили, чтобы подстроить отдельные части формулы. Напомним также,

что команды `\tfrac` и `\dfrac` (см. разд. 4.1.2), генерирующие дроби стилей `\textstyle` и `\displaystyle`, позволяют удобно управлять размером символов числителя и знаменателя.

Разберем использование математических стилей на примере дроби, стоящей в показателе экспоненты:

$$\mathrm{e}^{-\frac{\varepsilon}{kT}} \rightsquigarrow e^{-\frac{\varepsilon}{kT}}.$$

Стоящая в числителе буква ε оказывается слишком малой при размере `SS`. Используя команду `\tfrac`, все символы дроби можно увеличить до размера `S`: $e^{-\frac{\varepsilon}{kT}}$, но тогда формула раздвигает строки, и знаменатель дроби выглядит слишком большим. Формула станет гораздо лучше, если подстроить только размер символа ε : $\mathrm{e}^{-\frac{\scriptstyle\varepsilon}{kT}}$.

Обратите внимание, что число $e = 2.71828\dots$, как и все другие константы, принято набирать шрифтом `\mathrm`, так как оно не является переменной « e », например зарядом электрона.

4.1.6. Пробелы в формулах

Обычные пробелы, набираемые с клавиатуры, в формулах полностью игнорируются, ввести их можно только с помощью команд. Для пробелов фиксированной ширины используются все команды, перечисленные в табл. 2.3. Команда «`\`» вставляет пробел, ширина которого может немного меняться. Команды

$$\hspace{\text{длина}} \quad \text{и} \quad \mspace{\text{длина}}.$$

позволяют регулировать ширину пробела. В первой из них длина может быть задана в любых единицах, а во второй — только в единицах `\mu = (1/18)em`. Как и в тексте, положительная длина раздвигает элементы формулы, а отрицательная — их сближает.

В тонкой настройке формулы помогают «фантомы», незримо участвующие в вычислении ее размеров. Аргументами команд-фантомов служат математические выражения:

$$\hphantom{\dots}; \quad \vphantom{\dots}; \quad ,$$

из которых `\hphantom` создает пробел с шириной выражения; `\vphantom` вставляет в формулу элемент нулевой ширины, имеющий высоту и глубину выражения. И наконец, `\phantom` вставляет пустой бокс с полными размерами выражения. Отметим, что все три команды могут использоваться не только в формулах, но и в тексте. В этом случае их аргументом является часть строки текста.

Поясним использование фантомов следующим примером. Сумма корней $\sqrt{a} + \sqrt{d} \sqrt{y}$ $\rightsquigarrow \sqrt{a} + \sqrt{d} + \sqrt{y}$ смотрится лучше, если их размеры одинаковы:

$$\sqrt{a\vphantom{dy}} + \sqrt{d\vphantom{y}} + \sqrt{y\vphantom{d}} \rightsquigarrow \sqrt{a} + \sqrt{d} + \sqrt{y}.$$

Это обеспечивают фантомы, корректирующие высоту и глубину подкоренных выражений. Результат достигается с помощью довольно громоздких конструкций, которые можно заменить одной командой `\mathstrut` $\equiv \vphantom{\{}}$, вставляющей «распорку» с высотой круглой скобки:

$$\sqrt{a\mathstrut} + \sqrt{d\mathstrut} + \sqrt{y\mathstrut} \rightsquigarrow \sqrt{a} + \sqrt{d} + \sqrt{y}.$$

4.1.7. Текст в формулах

Русские буквы в формулах не определены, однако вставить в них текст очень просто. Для этого можно использовать любую команду установки параметров текстового шрифта из табл. 2.5, начинающуюся словом `text`, например:

$$abcd \neq \textit{абвг} \rightsquigarrow abcd \neq \textit{абвг}.$$

Специальная команда

`\text{текст}`

вставляет `текст`, применяя шрифт, которым набран текст перед формулой: $\{\textbf{\text{\textit{ë} x_{\text{ы}}}}\} \rightsquigarrow \textbf{\textit{ë}x_{\text{ы}}}$.

Вставить фразу, связывающую строки сложной формулы, позволяет команда

`\intertext{текст}`.

Она должна стоять после команды `\\` в начале следующей строки:

```
\begin{gather}
  x \equiv z, \\
\intertext{если}
  x=y \text{ и } y=z.
```

$$x \equiv z, \quad (4.1)$$

```

  x=y \text{ и } y=z.
\end{gather}
```

$$x = y \text{ и } y = z. \quad (4.2)$$

4.2. Математические блоки

В дополнение к перечисленным ранее основным элементам и конструкциям формул имеются математические блоки, позволяющие создавать выражения с очень сложной структурой даже в формулах, находящихся в строке. Например, формулу

$$|\sin \alpha| = \begin{cases} \sin \alpha, & 0 \leq \alpha \leq \pi, \\ -\sin \alpha, & \pi \leq \alpha \leq 2\pi, \end{cases}$$

можно сверстать, используя блок `cases`.

блок `cases`.

Компилятор обрабатывает блоки в математической моде, но сами они ее не иницируют и потому должны использоваться только внутри формул. Сколь бы сложную структуру не имел блок, он является всего лишь частью формулы, в которой стоит.

Блоки представляют собой окружения со структурой типа таблицы `tabular`. Каждая строка содержит выровненные поля (ячейки), которые разделяются символом `&`, а сами строки разделяются командами `\\[длина]`. Одинаковые поля разных строк образуют одинаково выровненный столбец (колонку) и связываются общей вертикалью выравнивания.

Используя положительную или отрицательную длину в параметре команды

$$\\[длина],$$

можно сдвигать и раздвигать строки. Следует помнить, что в математике команда `\\` определена только в части окружений, а ее использование вне этих окружений вызовет ошибку компиляции.

В приведенных далее примерах мы поместили столбцы и тип их выравнивания. Конструкцией «`\>|<`» обозначено центрирование.

Знаками « \rightarrow » « \leftarrow » показаны поля с правым и левым выравниванием. Общая вертикаль, образованная столбцами с правым и левым выравниванием, обозначена конструкцией « $\rightarrow\leftarrow$ ». Промежутки между столбцами, вычисляемые автоматически, обозначены отрезками « \mid », максимальные промежутки — конструкцией « $\left\langle \right\rangle$ », а промежутки, заданные автором, — отрезками « $\left\langle \rightarrow \right\rangle$ ». Символом & помечены все столбцы, кроме первого, который формируется автоматически.

Имена окружений, создающих математические блоки, перечислены в табл. 4.12.

Математические блоки Таблица 4.12

cases	aligned	array	matrix	bmatrix	vmatrix
split	gathered	smallmatrix	pmatrix	Bmatrix	Vmatrix

Простейшим блоком является окружение `gathered`, которое содержит набор центрированных строк. В контексте описанной ранее аналогии его можно рассматривать как таблицу с одной центрированной колонкой. Строки верстаются в стиле `\displaystyle`:

<code>gathered</code>	— набор центрированных строк
-----------------------	------------------------------

<code>\[\begin{gathered}</code>	$x + y = a + c$
<code>x+y = a+c \\\</code>	$z = x + d$
<code>z = x+d</code>	$\left\langle \right\rangle$
<code>\end{gathered}\]</code>	

Большой набор блоков составляют матрицы, верстаемые в виде таблиц с несколькими центрированными столбцами, разделенными небольшими одинаковыми промежутками.² Поля набираются в стиле `\textstyle`:

² Счетчик `MaxMatrixCols` хранит максимальное число колонок матриц. По умолчанию его значение равно десяти.

Матрицы: `matrix`, `pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix`, `Vmatrix`

<code>\[\begin{matrix}</code>	$\begin{matrix} x & a+b \\ 0 & y \end{matrix}$	<code>\]</code>	<code>\[\begin{pmatrix}</code>	$\begin{pmatrix} x & a+b \\ 0 & y \end{pmatrix}$	<code>\]</code>
<code>\[\begin{bmatrix}</code>	$\begin{bmatrix} x & a+b \\ 0 & y \end{bmatrix}$	<code>\]</code>	<code>\[\begin{Bmatrix}</code>	$\begin{Bmatrix} x & a+b \\ 0 & y \end{Bmatrix}$	<code>\]</code>
<code>\[\begin{vmatrix}</code>	$\begin{vmatrix} x & a+b \\ 0 & y \end{vmatrix}$	<code>\]</code>	<code>\[\begin{Vmatrix}</code>	$\begin{Vmatrix} x & a+b \\ 0 & y \end{Vmatrix}$	<code>\]</code>

Как правило, матрицы, имеющие большой размер, размещаются в вынесенных формулах. Для верстки матриц в строчных формулах имеется окружение `smallmatrix`, использующее мелкий шрифт и малые промежутки между столбцами и строками. Например, определитель $\begin{vmatrix} x & 0 \\ 1 & y \end{vmatrix}$ получен следующим способом:

```
\left| \begin{smallmatrix} x & 0 \\ 1 & y \end{smallmatrix} \right|
```

Блок `array` позволяет верстать сложные конструкции с произвольным количеством полей и любым типом выравнивания. Окружение `array` является полным аналогом окружения `tabular`, оно имеет аргумент, содержащий последовательность столбцов, которые можно отчеркивать и т. д., как описано в разд. 3.2. Более того поля (ячейки) можно объединять, используя команды `\multicolumn` и `\multirow`. Хотя сам блок `array` форматируется в математической моде в стиле `\textstyle`, столбцы «р» и поля, объединенные с помощью команды `\multirow`, заполняются в текстовой моде. Как показано в примере, с помощью окружения `array` можно сделать матрицу с разным выравниванием столбцов, а конструкция `\left•` и `\right•` позволяет охватить ее скобками:

array — блок с произвольным выравниванием столбцов

```

\left(
  \begin{array}{lcr}
    x & a+b & c+d \\
    e+f & y & 0 \\
    \multicolumn{3}{c}{\dotfill} \\
    1 & 0 & z \\
  \end{array}
\right)

```

$$\left(\begin{array}{lcr} x & a+b & c+d \\ e+f & y & 0 \\ \multicolumn{3}{c}{\dotfill} \\ 1 & 0 & z \end{array} \right)$$

Блок `cases` предназначен для записи вариантов соотношений, объединенных фигурной скобкой. Он имеет два поля с левым выравниванием. Первое, прижатое к скобке, предназначено для записи выражения, а второе — для записи условия. Два столбца автоматически разделяются небольшим пустым промежутком. Поля набираются в стиле `\textstyle`:

cases — блок с выравниванием и объединяющей скобкой

```

\[ x =
  \begin{cases}
    y, & y \leq 0, \\
    (y/a)^2, & y \geq 0.
  \end{cases}
\]

```

$$x = \begin{cases} y, & y \leq 0, \\ (y/a)^2, & y \geq 0. \end{cases}$$

Блок `split` позволяет расщепить длинные строки, или составить систему уравнений. Он имеет одну вертикаль выравнивания, к которой прижимаются левые и правые поля столбцов. Поля набираются в стиле `\displaystyle`:

split — блок с общей вертикалью выравнивания

```

\[ \begin{split}
  x &= y+z \\
  y-z &= a+b+z^2
\end{split} \]

```

$$\begin{aligned} x &= y+z \\ y-z &= a+b+z^2 \end{aligned}$$

В окружении `aligned` поля определены циклически: поле с правым выравниванием, поле с левым выравниванием, снова поле с правым выравниванием и т. д. Количество полей не ограничено. Соседние столбцы с правым и левым выравниванием связаны общей вертикалью, а между столбцами с левым и правым выравниванием вставляется небольшой промежуток. Поля верстаются в стиле `\displaystyle`:

`aligned` — блок с произвольным числом вертикалей выравнивания

```
\begin{aligned}
x &= y+z & y &= a+z & \bullet & \bullet & \dots \\
\end{aligned}
\qquad \dots
```

$$\begin{array}{ccc}
& \& & \& \& & \& \& \\
& \rightarrow \parallel \leftarrow & \vdash & \rightarrow \parallel \leftarrow & \vdash & \rightarrow \parallel \leftarrow \\
x = y + z & & y = a + z & & \bullet \bullet \dots \\
a + b = y & & c + d = z & & \bullet \bullet \dots
\end{array}$$

4.3. Нумерованные формулы

4.3.1. Нумерация формул

Для верстки вынесенных формул используются окружения, перечисленные в табл. 4.13.

Математические окружения

Таблица 4.13

<code>equation</code>	<code>eqnarray</code>	<code>align</code>	<code>alignat</code>
<code>multline</code>	<code>gather</code>	<code>flalign</code>	<code>xalignat</code>

Созданные с их помощью формулы автоматически нумеруются, а для ссылок на них имеется команда `\eqref{метка}`, выводящая номер формулы, заключенный в круглые скобки. Чтобы сослаться на формулу, ее нужно пометить командой `\label{метка}`. Если окружение содержит одну формулу, метку можно поставить как перед ней, так и после нее. В окружениях, содержащих несколько формул, пометить следует все, поставив команды

`\label` в начале или конце формул, которые разделяются командами `\\`. В последней строке команду `\\` использовать не стоит, иначе она создаст пустую формулу.

Покажем, что окружению, содержащему несколько нумерованных формул, собственный номер не присваивается.

```
\begin{eqnarray}\label{e:array}
```

$$x \ \&\& \ y+a \ \label{e:x} \\ x = y + a \quad (4.3)$$

$$z \ \&\& \ a+x \ \label{e:z} \quad z = a + x \quad (4.4)$$

```
\end{eqnarray}
```

Сошлемся на «систему» `\eqref{eq:array}` \rightsquigarrow (4.3) и оба ее уравнения: `\eqref{eq:x}` \rightsquigarrow (4.3), `\eqref{eq:z}` \rightsquigarrow (4.4). Как видно из ссылок, каждая формула имеет свой номер, а метки `eq:sample` и `eq:x` ассоциированы с первым уравнением, так как обе находятся в одной строке-формуле.

Поставив вместо метки команду `\nonumber` или `\notag`, нумерацию отдельной формулы можно отменить, а чтобы подавить нумерацию всех формул, в имя окружения нужно добавить звездочку. Все перечисленные в табл. 4.13 окружения имеют вариант имен со звездочкой. Если в предыдущем примере использовать окружение `eqnarray*`, строки уравнений нумероваться не будут.

Чтобы присвоить номера отдельным строкам системы уравнений и системе в целом, используется окружение `subequations`. В него помещается метка, служащая для ссылки на общий номер системы, а также одно или несколько окружений с формулами и собственными метками. Получив текущий номер, окружение `subequations` присвоит его всем формулам и дополнительно помечит их разными буквами. Например, система (4.5) состоит из уравнений (4.5a) и (4.5b):

```
\begin{subequations}\label{e:sample2}
```

```
\begin{eqnarray}
```

$$x \ \&\& \ y+a \ \label{e:x2} \\ x = y + a \quad (4.5a)$$

$$z \ \&\& \ a+x \ \label{e:z2} \quad z = a + x \quad (4.5b)$$

```
\end{eqnarray}
```

```
\end{subequations}
```

Для тонкой настройки номеров предусмотрены команды

`\tag{текст}` и `\tag*{текст}`

Обе они печатают текст в поле, предназначенном для номера формулы, но первая заключает его в скобки, а вторая — нет. Они нужны, чтобы заменить номер, проставляемый автоматически, или пометить формулы в уравнениях, создаваемых с помощью конструкций `\[... \]` и окружений «со звездочкой», в которых нумерация отключена. При их использовании значение счетчика формул не меняется. Приведем пример из книги [3]:

<code>\begin{equation}\label{e:a-b}</code>	На обе формы уравнения
<code> a - b = 0,</code>	$a - b = 0,$ (4.6)
<code>\end{equation}</code>	$a = b,$ (4.6')
<code>\begin{equation}\label{e:a=b}</code>	можно сослаться независимо:
<code> a = b. \tag{\ref{e:a-b}},\,\$'\\$}</code>	<code>\eqref{e:a-b} \rightsquigarrow (4.6),</code>
<code>\end{equation}</code>	<code>\eqref{e:a=b} \rightsquigarrow (4.6').</code>

4.3.2. Структура формул

Перейдем к анализу конструкций, применяемых для верстки нумерованных формул.

Окружения `equation` и `multiline` служат для набора отдельных формул, которым присваивается только один номер, хотя содержащиеся в них выражения могут состоять из нескольких строк. В `multiline` возможность разбиения формулы на несколько строк заложена в самом окружении, а в случае `equation` для этого можно использовать блоки.

Приведем пример окружения `equation` с блоком `cases`:

`equation` — нумерованная формула

```
\begin{equation}\label{•}
  |\sin x|=
  \begin{cases}
    \sin x, & 0 < x < \pi, \\
    -\sin x, & \pi < x < 2\pi.
  \end{cases}
\end{equation}
```

Окружение `multline` предназначено для верстки длинных выражений. Оно может содержать несколько строк, первая из которых прижимается влево, последняя — вправо, а остальные центрируются. Номер формулы проставляется в конце последней строки:

`multline` — формула с продолжением

<pre>\begin{multline} y = 2y+x+z- \\ -(y+z)-(y+x)+ \\ +(y+x)+y- \\ -(y+x)\label{\bullet} \end{multline}</pre>	$ \begin{array}{l} \leftarrow \\ y = 2y + x + z - \\ - (y + z) - (y + x) + \\ + (y + x) + y - \\ \triangleright\triangleleft - (y + x) \end{array} \quad (4.8) $
	$\rightarrow $

Окружение `gather` предназначено для набора нескольких центрированных формул:

`gather` — набор центрированных формул

<pre>\begin{gather}\label{\bullet} x+y = a+c \\ z = x+d \label{\bullet} \end{gather}</pre>	$x + y = a + c \quad (4.9)$ $z = x + d \quad (4.10)$
	$\triangleright\triangleleft$

Окружения, обсуждаемые далее, имеют строение аналогичное строению таблиц `tabular`. Каждая формула формально представляет собой строку таблицы и содержит выровненные поля (ячейки), которые разделяются символом `&`. Аналогично строкам таблиц, формулы, разделяемые командой `\\[длина]`, можно сдвигать и раздвигать по вертикали, используя положительную или отрицательную длину, однако в отличие от ячеек таблиц, поля формул нельзя объединять. Одинаковые поля разных формул образуют столбец (колонку), связанный общей вертикалью выравнивания.

В окружении `eqnarray` предусмотрено три поля: знаки отношений разных уравнений составляют центрированный столбец, а части выражений прижимаются к нему слева и справа, образуя

столбцы с правым и левым выравниванием. Части уравнения верстаются в стиле `\displaystyle`, а центральное поле — в стиле `\textstyle`. Для разделения полей достаточно двух символов `&`, поэтому попытка добавить еще один приведет к ошибке:

`eqnarray` — набор уравнений с тремя полями

«выравнивание справа – центрирование – выравнивание слева»

<code>\begin{eqnarray}\label{•}</code>	<code>& &</code>	
<code>x &=& a+z \\\</code>	$\rightarrow > < \leftarrow$	
<code>xy &=& a+b+z^2 \label{•} \\\</code>	$x = a + z$	(4.11)
<code>xyz &=& a-b+x^2 \label{•}</code>	$xy = a + b + z^2$	(4.12)
<code>\end{eqnarray}</code>	$xyz = a - b + x^2$	(4.13)

Окружение `align` аналогично блоку `aligned`. Поля в нем определены циклически: два соседних столбца с правым и левым выравниванием связываются общей вертикалью и т. д. Количество полей (столбцов) не ограничено. Соседние столбцы, не связанные общей вертикалью, автоматически разделяются промежутком:

`align` — набор уравнений с последовательностью полей

«правое – левое – вычисляемый промежуток»...

<code>\begin{align}\label{•}</code>				
<code>x & = y+z & & y & = a+z & & • & & • & \dots \\\</code>				
<code>\end{align} &</code>	\dots	<code>& &</code>	<code>& &</code>	<code>& &</code>
$\rightarrow \leftarrow$	-----	$\rightarrow \leftarrow$	-----	$\rightarrow \leftarrow$
$x = y + z$		$y = a + z$	$\bullet \bullet \dots$	(4.14)
$a + b = y$		$c + d = z$	$\bullet \bullet \dots$	(4.15)

В окружении `align` все выражения, в том числе и крайние, окружены слева и справа одинаковыми свободными полями, это приводит к центрированию уравнений на странице. В окружении `flalign` поля между выражениями имеют максимальную ширину, а крайние выражения прижимаются к краю страницы и номеру формулы:

flalign — набор уравнений с последовательностью полей «правое – левое – максимальный промежуток»...

```
\begin{flalign}\label{•}
      x = & y+z & & y = & a+z & & • & • & \dots \\
\end{flalign}
```

$$\begin{array}{rcccl}
 \rightarrow \parallel \leftarrow & & \leftarrow & & \rightarrow \parallel \leftarrow \\
 x = y + z & & y = a + z & & \dots \quad (4.16) \\
 a + b = y & & c + d = z & & \dots \quad (4.17)
 \end{array}$$

Если в окружениях `align` и `flalign` ширина пустых полей между выражениями вычисляется автоматически, то в окружении `alignat` она задается автором. Данное окружение имеет аргумент, указывающий количество вертикалей выравнивания. При его значении, равном n , строка содержит n вертикалей выравнивания и $n - 1$ пустых полей, ширина которых по умолчанию равна нулю. Чтобы изменить ее, рядом с четными символами `&` нужно поставить команду-пробел нужной ширины:

alignat — набор уравнений с последовательностью полей «правое – левое – заданный промежуток»...

```
\begin{alignat}{3}\label{•}
      x = & y+z & & \llap{\quad} c+d = & a+z & & • & • & \dots \\
\end{alignat}
```

$$\begin{array}{rcccl}
 \rightarrow \parallel \leftarrow & & \leftarrow \longrightarrow & & \rightarrow \parallel \leftarrow & & \rightarrow \parallel \leftarrow \\
 x = y + z & & c + d = a + z & & \dots \quad (4.18) \\
 a + b = y & & y = z & & \dots \quad (4.19)
 \end{array}$$

Окружение `xalignat`, позволяет регулировать ширину промежутка между выражениями и при этом заботится, чтобы она не

была нулевой. Длина пробела, заданного вручную, добавляется к ширине, вычисленной автоматически. Окружения `xalignat` и `alignat` имеют одинаковый синтаксис:

`xalignat` — набор уравнений с последовательностью полей «правое – левое – вычисляемый + заданный промежуток»...

```
\begin{xalignat}{3}\label{•}
  x = & y+z & \quad c+d = & a+z & \quad • & \quad • & \quad \dots \\
\end{xalignat}
      \dots
      \begin{array}{ccc}
        \begin{array}{c} \rightarrow \parallel \leftarrow \\ x = y + z \end{array} & \begin{array}{c} \leftarrow \parallel \rightarrow \\ c + d = a + z \end{array} & \begin{array}{c} \leftarrow \parallel \rightarrow \\ \bullet \bullet \dots \end{array} \\
        a + b = y & y = z & \bullet \bullet \dots \end{array} \quad (4.20)
      \end{array} \quad (4.21)
```

Уравнения, состоящие из нескольких строк, помещаются на страницу целиком. Среди перечисленных окружений только `eqnarray` позволяет автоматически разбивать их между страницами. В остальных случаях на это нужно разрешение автора. С этой целью пакет `amsmath` вводит команду

`\displaybreak[число]` .

Число, изменяющееся в пределах от 0 до 4, регулирует «жесткость» указания компилятору от рекомендации (0), до обязательного выполнения (4), используемого по умолчанию. Команду `\displaybreak` обычно ставят непосредственно перед командой перевода строки `\\`.

Запрет на разбиение уравнений можно снять, добавив в преамбулу документа команду

`\allowdisplaybreak[число]` .

Ее параметр регулирует действия компилятора аналогично параметру команды `\displaybreak`. При этом, если автоматический перенос части уравнения на другую страницу разрешен, команду `*` можно использовать для группирования строк уравнений путем их «склеивания» (см. с. 47).

4.4. Верстка сложных формул

Сложные формулы собираются подобно конструктору. Вначале выбирается оптимальное окружение (таблица), связывающее строки формул, затем, при необходимости, поля формул (ячейки таблицы) усложняются с помощью блоков. На конечном этапе отдельные части формул настраиваются так, чтобы вся конструкция имела оптимальный вид.

Поясним сказанное на примере сложного выражения:

$$I + \delta I = \begin{cases} I_c[2f(r, b) - f(r, a)], & r \leq a \\ I_c[2f(r, b) - 1], & a \leq r \leq b, \\ I_c, & b \leq r \leq R, \end{cases} \quad (4.22)$$

$$\frac{B_z + \delta B_z}{\mu_0} = \begin{cases} H + I_c \left[P(r, a) - \frac{1}{2 \operatorname{ch}(R/r)} \right], & a \leq r \leq b, \\ H - \delta H + I_c \left[\frac{1}{2 \operatorname{ch}(R/r)} + \right. \\ \quad \left. + P(r, a) - 2P(r, b) \right], & B \leq r \leq R. \end{cases} \quad (4.23)$$

Оно сверстано с помощью следующего кода:

```
\begin{eqnarray}\label{eq:I-hysteresis}
I+\delta I
&=&
\begin{cases}
I_c[2f(r,b)-f(r,a)], & & r\leq a\\
I_c[2f(r,b)-1], & & a\leq r\leq b,\\
I_c, \hphantom{H+I_c\left[P(r,a)-\right.} & & \left.\dfrac{1}{2\operatorname{ch}(R/r)}\right]} \\
& & & & b\leq r\leq R,
\end{cases}
\end{eqnarray}
\frac{B_z+\delta B_z}{\mu_0}
&=&
\begin{cases}
H+I_c\left[P(r,a)-\dfrac{1}{2\operatorname{ch}(R/r)}\right], \\
\hphantom{I_c} & & \{a\}\leq r\leq b,
\end{cases}
```

```

\begin{split}
H-\delta H
& +I_c\biggl[\frac{1}{2}\ch(R/r)+\ll[-1ex]
& +P(r,a)-2P(r,b)\biggr],
\end{split}
& B\leq r\leq R.
\end{cases}\label{eq:Bz-hysteresis}
\end{eqnarray}

```

Основной конструкцией, связывающей уравнения для тока I и индукции поля B_z , является окружение `eqnarray`, формирующее формулы (4.22) и (4.23). Гистерезис тока и поля описывается набором условий, оформленных с помощью блоков `cases`.

В первом выражении для B_z использована дробь `\dfrac`, так как в стиле `\textstyle`, которым набираются строки окружения `cases`, дробь смотрелась бы не очень хорошо. Второе выражение, имеющее слишком большую длину, разбивается блоком `split`. Данное окружение верстается в стиле `\displaystyle`, поэтому дробь в нем набрана с помощью обычной команды `\frac`. Так как большие скобки излишне раздвигают строки выражения, находящегося в блоке `split`, межстрочный интервал уменьшен с помощью команды `\ll[-1ex]`

Конструкции `cases` в разных строках формулы между собой никак не связаны, однако желательно, чтобы столбцы их условий имели общую вертикаль выравнивания. Это можно достичь, сделав одинаковой ширину столбцов значений. С этой целью в поле значений, имеющее минимальную длину в первом окружении, добавлен горизонтальный фантом (`\hphantom{\bullet}`) самого длинного выражения из второго окружения; в свою очередь, в длинное поле второго окружения добавлен самый короткий фантом из первого. Таким образом, длины полей выровнялись. Аналогичный эффект можно получить и проще, поставив в конце короткого выражения пробел подобранной длины, но в данном случае использование фантомов представляется вполне оправданным, так как оно автоматически гарантирует одинаковую ширину обоих столбцов.

4.5. Теоремы

Математика — наука строгая, требующая доказательства выдвинутых утверждений. Их формы могут быть очень разнообразны, например аксиома, теорема, лемма, следствие и т. д. Некоторые из них, как аксиома, не доказываются. Доказательство других может быть очень коротким, или, напротив, занять несколько страниц. В ходе доказательства могут возникать новые утверждения, которые тоже нужно доказывать. Возникающие таким образом иерархические структуры утверждений и доказательств невозможно свести к какой-то единой, наперед заданной форме, поэтому \LaTeX дает авторам возможность строить их самостоятельно. Для этого мы сами должны выработать формы утверждений и их подчиненность, создавая для них новые окружения.

Будем использовать термин теорема как синоним всех форм утверждений и опишем средства их формирования, предоставляемые ядром \LaTeX и стандартным пакетом `amsthm` из коллекции `amslatex` Американского математического общества. Ресурсы пакета `amsmath` специально выделяться не будут, поэтому подчеркнем, что в документах, содержащих теоремы, его нужно добавить в список загружаемых пакетов.

Простейшую форму утверждения вводит команда:

```
\newtheorem*{имя}{форма утверждения}.
```

Она создает окружение, имя которого должно быть уникальным. Второй аргумент команды задает заголовок, выводимый перед утверждением, формируемым при использовании окружения. Поясним сказанное примером:

```
\newtheorem*{theorem}{Теорема}
...
\begin{theorem}[Ферма]           Теорема (Ферма). Уравнение
...                                $m^n + l^n = k^n$  не имеет решения
\end{theorem}                   для целых чисел  $n > 2$ ,  $m$ ,  $k$ ,  $l$ .
```

Создаваемое окружение имеет параметр, позволяющий уточнить форму утверждения. Чаще всего в нем, как в приведенном примере, указывается автор теоремы.

Для оформления доказательств предназначено окружение:

```
\begin{proof}[заголовок] .
```

По умолчанию, в начале доказательства выводится заголовок *Доказательство*, в его конце автоматически ставится знак \square , означающий, «что и требовалось доказать». Параметр окружения позволяет заменить стандартный заголовок любым другим, или изменить его шрифт. Окружение `proof` используется для простых доказательств, не требующих дополнительных утверждений:

```
\begin{theorem}
```

Волга впадает в Каспийское море.

Теорема. *Волга впадает в Каспийское море.*

```
\end{theorem}
```

```
\begin{proof}
```

Смотри географическую карту России.

Доказательство.

Смотри географическую карту России. \square

```
\end{proof}
```

Длинные доказательства со сложной структурой можно разбить на части командами разбивки, описанными в разд. 5.2.

Еще одну форму теорем вводит команда:

```
\newtheorem{имя}[счетчик]{форма утверждения} .
```

Она не только определяет новое окружение, но и ассоциирует с ним счетчик, получающий такое же имя. Теоремы получают номер, а метка, поставленная внутри окружения, позволяет сослаться на него. Параметр команды `\newtheorem` дает возможность использовать уже существующий счетчик,³ что позволяет вести общий подсчет нескольких форм утверждений, например постулатов и аксиом:

³ При этом счетчик с именем окружения тоже вводится и отождествляется с существующим счетчиком.


```

\newtheorem{axiom}{Аксиома}
\newtheorem{postulate}[axiom]{Постулат}
...
\begin{postulate}[для атеистов]
  Бога нет! \label{•} Постулат 1 (для атеистов).
\end{postulate}
Бог не!
\begin{axiom}[для верующих]
  Бог есть! \label{•} Аксиома 2 (для верующих).
\end{axiom}
Бог есть!

```

Так как номер разделяет заголовок утверждения и его уточнение, декларация `\swapnumbers` помогает изменить порядок вывода номера. Ее нужно использовать до определения теорем командами `\newtheorem`. Например:

```

\swapnumbers
\newtheorem{aphorism}{Афоризм}
...
\begin{aphorism}[Козьма...]
...
1 Афоризм (Козьма Прутков).
\end{aphorism}
Нельзя объять необъятное.

```

Нумерацию утверждений, возникающих в ходе доказательства основной теоремы можно согласовать с ее номером. Для этого имеется команда:

```

\newtheorem{имя}{форма утверждения}[счетчик] ,

```

в параметре которой указывается имя основной теоремы, номер которой выводится перед номером утверждения. При формировании новой теоремы такие утверждения подсчитываются заново. Данную команду также используют, чтобы нумеровать теоремы по главам или разделам. В этом случае в параметре указывается соответствующий счетчик, значение которого выводится перед номером теоремы.

Приведем еще один пример:

```

\swapnumbers
\newtheorem{mat}{Теорема}
\newtheorem{lemma}{Лемма}[mat]

```

```

...
\begin{mat}[материализм]
  Все родилось из ничего.
\end{mat}
...
\begin{lemma}
  Материя существует вечно.
\end{lemma}
...
\begin{lemma}
  Вечность -- это...
\end{lemma}

```

1 Теорема (материализм).
Все родилось из ничего.

1.1 Лемма. *Материя существует вечно.*

1.2 Лемма. *Вечность — это движение времени по замкнутой траектории.*

По умолчанию все утверждения оформляются одинаково. Декларация

```
\theoremstyle{стиль}
```

позволяет выбрать для них три стиля вывода, конкретная реализация которых зависит от класса документа. Стиль должен быть объявлен до определения окружений командами `\newtheorem`.

В стиле `plain`, используемом по умолчанию, в стандартных классах заголовки теоремы делается жирным, а ее текст набирается курсивом, как показано в примерах выше.

В стиле `definition` заголовок также жирный, а для текста используется обычный шрифт:

```

\theoremstyle{definition}
\newtheorem*{def}{Определение}

```

Определение. Интернет это виртуальная реальность, данная нам в искушение.

В стиле `remark` для заголовка выбран курсив, а для текста обычный шрифт:

```

\theoremstyle{remark}
\newtheorem*{rem}{Замечание}

```

Замечание. E pur si muove!
(Galileo Galilei)

Стандартный пакет `theorem` [16] вводит дополнительные стили оформления теорем, позволяет настроить шрифты заголовков и формулировок, а также их вертикальные отбивки.

Подводя итог, сделаем одно важное замечание. Команды `\newtheorem` являются глобальными декларациями, область действия которых не ограничивается какой-то группой. Определения теорем следует размещать в преамбуле. Приведем пример определения списка теорем:

```
\swapnumbers
\newtheorem{axiom}{Аксиома}
\newtheorem{theorem}{Теорема}
\newtheorem{lemma}{Лемма}[theorem]

\theoremstyle{definitin}
\newtheorem*{definition}{Определение}

\theoremstyle{remark}
\newtheorem{sequence}{Следствие}
    . . . .
\begin{document}
```

Для теорем, следствий и аксиом выбрана независимая нумерация, номера лемм подчинены номеру теоремы, а определения не нумеруются. Номера утверждений будут выводиться перед заголовками. Теоремы, леммы и аксиомы имеют стандартное шрифтовое оформление. Определения верстаются в стиле `definition`, а следствия — в стиле `remark`.

4.6. Коммутативные диаграммы

В математике, в частности в теории категорий, коммутативная диаграмма связывает объекты определенной категории путями — *морфизмами*. Для наглядности она изображается в виде структуры наподобие графа, вершинами которого служат объекты, а ребрами — морфизмы.

Для построения коммутативных диаграмм пакет `amscd` коллекции `amslatex` вводит окружение `CD`, которое является мате-

матическим блоком с особым синтаксисом. Рассмотрим его на следующем примере:

```

\begin{equation}
  \begin{CD}
    A @<<< B @>>> C @= D \\
    @| @AAA @VVV @. \\
    H @. F @((( G @))) E
  \end{CD}
\end{equation}

```

$$\begin{array}{ccccccc}
 A & \longleftarrow & B & \longrightarrow & C & \longequal{\quad} & D \\
 \parallel & & \uparrow & & \downarrow & & \\
 H & & F & \longleftarrow & G & \longrightarrow & E
 \end{array} \tag{4.24}$$

Диаграмма представляет собой прямоугольную сетку с узлами, связанными стрелками или знаками равенства. Узлы, помеченные в примере буквами, могут содержать любые выражения.

Сетка формируется строками двух типов. Одни состоят из узлов и горизонтальных связей, а другие — только из вертикальных связей. Если связь отсутствует, узлы разделяются командой @., необходимой для поддержания баланса узлов и связей.

Связи-морфизмы рисуют команды, собранные в табл. 4.14, имеющие специфический синтаксис. Их имена начинаются символом @, за которым следует символ, определяющий морфизм. В командах, рисующих стрелки, он указывает направление стрелки. Для вывода знаков горизонтального и вертикального равенства используются символы = и |.

Команды @AAA (↑) и @VVV (↓) рисуют вертикальные стрелки. В обозначениях горизонтальных стрелок использованы знаки сравнения и круглые скобки, например @<<< (←) и @))) (→). Открывающая скобка служит «синонимом» знаку меньше, а закрывающая — знаку больше.

Стрелки могут иметь индексы, которые ставятся между первой и второй парой символов, указывающих направление, напри-

мер: $\textcircled{>a>c} \rightsquigarrow \xrightarrow[c]{a}$. Сложные индексы лучше заключить в фигурные скобки: $\textcircled{>\{g\circ h\}} \rightsquigarrow \xrightarrow{g\circ h}$. Таким же образом следует поступить, если индекс содержит символ, используемый в имени команды: $\textcircled{>\{>\}} \rightsquigarrow \xrightarrow{>}$.

Морфизмы коммутативных диаграмм

Таблица 4.14

Команды	Примеры
$\textcircled{\cdot}$	AB
$\textcircled{=}$	$A \xlongequal{\quad} B$
$\textcircled{>\{вверху\}>\{внизу\}}$ $\textcircled{\{вверху\}\{внизу\}}$	$A \xrightarrow[b]{a} B$
$\textcircled{<\{вверху\}<\{внизу\}<}$ $\textcircled{\{вверху\}\{внизу\}\{}$	$A \xleftarrow[b]{a} B$
$\textcircled{\cdot}$	$A \quad A \quad A \quad A$
$\textcircled{ }$	$\parallel \quad a \uparrow b \quad a \downarrow b$
$\textcircled{A\{слева\}A\{справа\}A}$	
$\textcircled{V\{слева\}V\{справа\}V}$	$B \quad B \quad B \quad B$

Ресурсов окружения CD недостаточно для создания сложных диаграмм с разными типами связей и сложной логикой, в которых объекты могут иметь большое число связей. С этой целью лучше использовать пакет `xu` из коллекции `XU-pic`, предоставляющий самые широкие возможности, включающие дополнительные (диагональные) связи и генерацию любого типа стрелок с управляемой кривизной. Его оригинальные команды и конструкции имеют довольно сложный синтаксис, подробно описанный в документации пакета [17].

Глава 5

Библиография и списки

Как правило, рукопись содержит какой-то список. Это может быть библиография к статье, предметный указатель в книге, или просто перечисление чего-либо. \LaTeX располагает обширным набором средств оформления таких списков.

5.1. Списки в тексте

Списки, используемые непосредственно в тексте, делятся на три типа: нумерованные, ненумерованные и перечисление описаний каких-то параметров, терминов, понятий и т. д. Они создаются с помощью окружений `enumerate`, `itemize` и `description`, имеющих одинаковый синтаксис, проиллюстрированный следующим примером нумерованного списка:

```
\begin{enumerate}
  \item Текст первого пункта...
    Второй абзац первого пункта.
  \item Текст второго пункта...
    Второй абзац второго пункта.
  \item И так далее...
\end{enumerate}
```

1. Текст первого пункта...
Второй абзац первого пункта.
2. Текст второго пункта...
3. И так далее...

Списки делятся на элементы, или записи, и могут содержать любое их количество. В свою очередь записи могут состоять из нескольких абзацев текста. В тексте программы каждая запись начинается командой

```
\item[настройка метки]
```

В сверстанном документе элемент помечается меткой, которая автоматически ставится перед ним. Вид меток определяется типом списка.

Ненумерованные списки формирует окружение

```
\begin{itemize},
```

пример его использования можно увидеть в разд. 3.1.1. Записи помечаются символами-метками, набор которых определяется классом документа. В стандартных классах записи основного списка помечаются символом «•», а записи вложенных списков символами «-», «*» и «·». Параметр команды `\item` позволяет заменить метку отдельной записи. Например, запись `\item[+]...` будет помечена знаком «+». Глобальная замена меток описана в разд. 9.7.1.

Списки любых типов можно вкладывать друг в друга, при этом глубина вложенности списков одного типа не должна превышать четырех уровней, а общая вложенность не может быть больше шести. Вместе с изменением уровня вложенности изменяются отбивки и метки записей.

Используем окружение `itemize`, чтобы показать пример организации вложенных списков:

```
\begin{itemize}
  \item Первая запись...
  \item Его вторая запись.
    \begin{itemize}
      \item Первая запись...
      \item Его следующая запись.
    \end{itemize}
  \item Вернулись из...
\end{itemize}
```

- Первая запись списка.
- Его вторая запись.
 - Первая запись вложенного списка.
 - Его следующая запись.
- Вернулись из вложенного списка в основной.

Нумерованные списки верстает окружение

```
\begin{enumerate }
```

Элементам этого списка присваивается номер. С каждым уровнем вложенности ассоциирован свой счетчик записей. Формат вывода счетчика зависит от вложенности списка. Приведем пример вложенных списков:

1. Элементы основного списка `enumerate` нумеруются арабскими цифрами.
 - (a) Списки второго уровня — прописными латинскими буквами.
 - i. Третьего уровня — строчными римскими цифрами.
 - A. Список четвертого уровня нумеруется заглавными латинскими буквами.
2. При возврате на прежний уровень нумерация продолжается.

Номер элемента можно заменить другой меткой. Для этого укажите ее в параметре команды `\item`. «Пропущенный» номер, получит следующий элемент. Для подавления вывода номера используйте пустой параметр.

На номер элемента можно сослаться командой `\ref`, если в нем поставить метку `\label`. Например, последний пункт вложенного списка имеет номер 1(a)iA.

Для верстки списка терминов, параметров и т. д. предназначено окружение

```
\begin{description}
```

использованное в описаниях классов документов (см. разд. 1.4.1), пакетов и их настроек (см. разд. 3.1.2), стилей и баз BibTeX разд. 5.2.3.

Предполагается, что в таком списке записи имеют заголовок и поясняющий текст. Заголовок, помещаемый в параметр команды

`\item[заголовок]`,

по умолчанию выделяется жирным шрифтом. В параметре можно указать команды, изменяющие стиль его печати:

```
\begin{description}
  \item[Раки]...
  \item[\it Жабь]...
\end{description}
```

Раки обитают в водоемах, любят чистую воду. Бывают маленькими, большими и ну о-о-о-очень большими.

Жабь обитают в болотах, водоемах и прочих сырых местах. Бывают зелеными и не очень.

В отличие от `itemize` и `enumerate`, вложенность списков `description` ограничена не четвертым, а шестым уровнем.

5.2. Список литературы

Формируя список литературы, или библиографию, автор сталкивается с двумя принципиальными трудностями.

Во-первых, полиграфические традиции издательств определяют большое разнообразие стилей оформления библиографии, поэтому список литературы, подготовленный для какого-то издания, для другого, скорее всего, придется верстать заново. Верстка же даже небольшой библиографии требует напряженного внимания и занимает значительное время.

Во-вторых, чаще всего библиография представляет собой нумерованный список. В рукописи источники должны нумероваться в порядке появления, и при этом их номера должны соответствовать порядку следования источников в библиографии. В процессе работы над рукописью порядок цитирования меняется очень часто, и нет смысла каждый раз приводить в соответствие ему список литературы. Однако даже по окончании редактирования сортировку библиографии, насчитывающей несколько десятков источников, сделать непросто. Чтобы обойти эту проблему в обзорных статьях, цитирующих большое количество работ, источники часто сортируются по фамилии первого автора, используемой в качестве ссылки в тексте рукописи.

Обе проблемы эффективно решаются с помощью специальной программы, формирующей список литературы, в качестве которой обычно выступает компилятор BibTeX. Для него уже накоплены большие ресурсы, кардинально упрощающие верстку библиографии, поэтому мы разберем метод ее формирования совместными усилиями компиляторов L^AT_EX и BibTeX.

Для начала покажем, как выглядят ссылка на источник и список литературы в программе:

.... в книге \cite{Knuth-TeXbook-1993} описаны ...

```
\begin{thebibliography}{99}
  \bibitem{Knuth-TeXbook-1993}
    Кнут~Дональд Е. \emph{Все про} \TeX --- Протвино: ...
```

Теперь покажем, как это выглядит в документе:

... в книге [1] описаны ...

Литература

[1] Кнут Дональд Е. *Все про T_EX* — Протвино: АО RD_TE_X, 1993 — С. 575. — ISBN: 5-900614-01-8

5.2.1. Библиография и цитирование

Список литературы формирует окружение

```
\begin{thebibliography}{текст}.
```

Текст в его аргументе определяет ширину поля, отводимого для меток. Для нумерованной библиографии в аргументе можно указать любое целое число с количеством цифр не меньшим, чем

у максимального номера в списке. Например, 99, если число источников не превышает сотни.

Записи списка начинаются командами

```
\bibitem[метка]{ярлык} ,
```

за которыми следует описание источника литературы. Если команда `\bibitem` имеет параметр, указанная в нем метка выводится вместо номера источника как в самом списке литературы, так и при ссылке на него в тексте.¹

В аргументе команды `\bibitem{•}` указывается ярлык записи, служащий для идентификации источника при цитировании. Каждый ярлык должен быть уникален.

Для цитирования предназначена команда

```
\cite[дополнительная информация]{список ярлыков} ,
```

В ее аргументе через запятую перечисляется список ярлыков. Элементы списка можно разделить пробелами, поставив их после запятой. Ставить же пробел после ярлыка не следует, т.к. он может вызвать сбой его идентификации. В программе `TeXstudio` при наведении курсора на ярлык всплывает окошко с описанием цитируемого источника, позволяющее проверить ссылку.

Отметим, что команда `\cite` является хрупкой (см. с. 34).

В процессе компиляции `LaTeX` находит записи с требуемыми ярлыками и, как показано в примере выше, заменяет команды `\cite` номерами (или метками) источников.

Команда `\cite` имеет параметр, позволяющий уточнить ссылку. Чаще всего в него помещают номера страниц, томов, разделов и т. д. Например, цитирование приведенного выше источника `\cite[c.125]{Knuth-TeXbook-1993}` в документе примет вид `[1, c.125]` и отошлет читателя к странице с нужной информацией в обширном руководстве по `TeX`.

Номера источников, перечисленных в аргументе команды `\cite`, не упорядочиваются. Чтобы сделать это, нужно загрузить пакет `cite`, который не только следит за их порядком, но и

¹ Мы полагаем, что библиография является нумерованным списком.

сокращает список последовательных номеров до диапазона. Пакет имеет гибкие настройки и вводит множество новых команд для управления всеми аспектами цитирования, подробно описанными в его документации [18].

Загрузка пакета `hyperref` преобразует ссылки на источники в гиперссылки. Вдобавок к этому, при использовании современных стилей верстки библиографии, например, из коллекций `gost`, `elsarticle`, `revtex`, адрес или название источника также преобразуются в гиперссылку, если его описание содержит адрес интернет ресурса.

5.2.2. Верстка библиографии с помощью BibTeX

Верстку библиографии следует доверить компилятору BibTeX. Для этого нужно создать библиографическую базу данных и обеспечить взаимодействие компиляторов L^AT_EX и BibTeX, блок-схема которого приведена на рисунке 5.1. Стрелками показаны направления потоков информации при взаимодействии компиляторов.

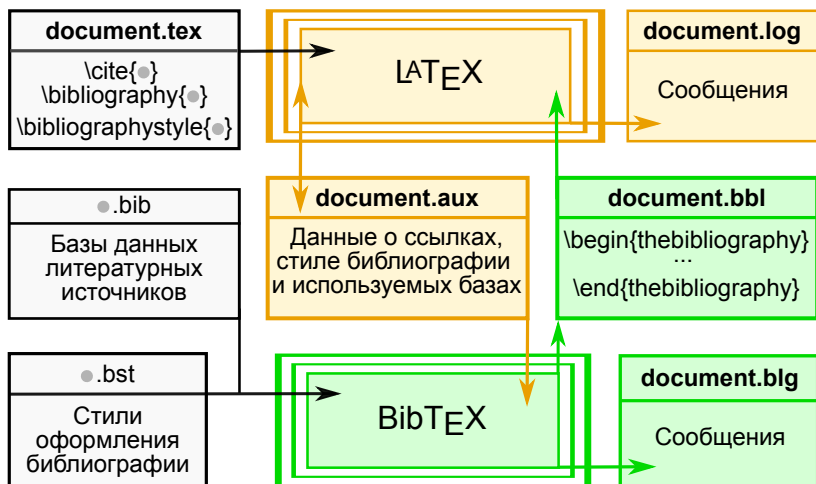


Рис. 5.1. Создание библиографии с использованием BibTeX

Рассмотрим пример, содержащий цитирование источников и команды, необходимые для взаимодействия компиляторов.

```
... \cite{Brandt-PRL-1993, Clem-PRB-1995}...  
  
\bibliographystyle{unsrt}  
\bibliography{Superconductivity, CriticalState}
```

Обнаружив команду `\cite{•}`, \LaTeX направляет содержащийся в ней список ярлыков в файл с расширением `•.aux`. Команды `\bibliographystyle{•}` и `\bibliography{•}` заносят в него информацию о стиле верстки библиографии и библиографических базах, которые нужно использовать для поиска источников. Порядок следования ярлыков, устанавливаемый в ходе компиляции, автоматически соответствует очередности их появления в командах `\cite`.

Запускаемый следом компилятор VibTeX , анализирует содержимое `•.aux`-файла, загружает указанный стиль и библиографические базы, извлекает из них записи с ярлыками цитируемых источников, формирует окружение `thebibliography` и помещает его в файл с расширением `•.bbl`.²

При последующих компиляциях \LaTeX загружает окружение `thebibliography` из этого файла и разрешает ссылки.

Если библиографию верстает VibTeX , для цитирования нужно использовать ярлыки, присвоенные источникам в библиографических базах. Если какой-то из ярлыков не найден, VibTeX выдаст сообщение об ошибке, направив его в файл с расширением `•.blg`. Сообщение об ошибке генерируется и в случае, если в источнике отсутствует информация, необходимая для правильного цитирования, например, название журнала или книги, год опубликования и т. д.

Для передачи списка библиографических баз компилятору VibTeX служит команда

```
\bibliography{список}.
```

² Напомним, что все автоматически генерируемые файлы наследуют имя компилируемого документа.

В ее аргументе через запятую перечисляются имена файлов, имеющих стандартное расширение `•.bib`, которое можно опустить. Данная команда замещается сверстанной библиографией, поэтому ее положение определяет место списка литературы в документе. Перед отправкой в редакцию на ее место нужно скопировать окружение `thebibliography` из `•.bbl`-файла, а саму команду следует закомментировать или убрать. Все необходимое для генерации списка литературы в издательстве должна содержать сама программа.

Стиль верстки библиографии задается командой

```
\bibliographystyle{установка стиля}.
```

Ее аргументом является имя загружаемого стилевого файла, имеющего расширение `•.bst`, которое также можно опустить. Перед отправкой в редакцию данную команду тоже нужно закомментировать или убрать.

Перечислим стандартные стили BibTeX, являющиеся основой других стилей.

plain Данный стиль верстает нумерованный список литературы, в котором источники сортируются в алфавитном порядке по имени первого автора, затем по годам и по названиям публикаций.

alpha Этот стиль выполняет сортировку аналогичную **plain**, но в библиографии и ссылках источники помечаются аббревиатурой, состоящей из первых букв фамилий трех первых авторов и последних двух цифр года выхода публикации, например, `JPG+05`. Знак `+` ставится, когда в публикации больше 3-х соавторов. Аббревиатуру единственного автора образуют первые три буквы фамилии.

abbrv Библиография верстается аналогично стилю **plain**, но имена авторов сокращаются до инициалов.

unsrt Данный стиль верстает список литературы, в котором источники не сортируются, а нумеруются в соответствии с порядком цитирования в программе.

Библиография документов, написанных по-русски, часто верстается в соответствии с ГОСТ 7.0.5-2008. В соответствии с ним список из четырех и менее соавторов предшествует названию публикации и выводится без сокращения, а более длинные списки, урезаются до первых трех соавторов и следуют за названием. Эти правила положены в основу стилей пакета `gost` :

`gost2008`, `ugost2008`, `gost2008l`, `ugost2008l` — стили, верстающие библиографию, в которой источники следуют порядку цитирования в рукописи, где стили с префиксом «u» предназначены для работы с библиографическими базами в формате `utf8`,³ стили с индексом «l» выводят полный список авторов перед названием публикации вне зависимости от их числа;

`gost2008s`, `ugost2008s`, `gost2008ls`, `ugost2008ls` — стили с индексом «s» сортируют источники либо по фамилии первого автора, либо по названию публикации, если число соавторов больше четырех.

Для правильной обработки программой записи русскоязычных источников в библиографических базах должны иметь поле `language = {russian}` (см. следующий раздел). Более подробную информацию о перечисленных стилях содержит описание пакета `gost`, подготовленное его автором И.А. Котельниковым [19].

При работе с библиографическими базами полезной оказывается команда

```
\nocite{список ярлыков} ,
```

которая заносит источники в список литературы, не вставляя при этом ссылку в документ. Использование звездочки в ее аргументе позволяет вывести все содержимое библиографических баз без явного перечисления ярлыков.

³ Эти стили предназначены для компиляторов, работающих с файлами в кодировке `utf8`, таких как `biber`. Программы `bibtex` и `bibtex8`, предназначенные для работы с 8-битными кодировками, могут неправильно сортировать источники в данной кодировке.

5.2.3. Синтаксис библиографических баз

Как правило, страничка статьи на сайте журнала или в библиографических базах типа SCOPUS или WEB OF SCIENCE содержит меню загрузки ее описания в формате BibTeX. Для работы с базами BibTeX имеется удобная свободно распространяемая программа **JabRef**. При создании новой записи она предлагает ввести DOI источника, а затем скачивает его описание из интернета.⁴ Если источником является книга, используя **JabRef**, можно сначала сделать новую запись типа **book**, затем в поле **ISBN**, находящемся в разделе **Optional fields**, ввести ISBN-код книги и, нажав иконку **Download**, расположенную справа от этого поля, загрузить описание записи.

Загрузка готовых записей дает наиболее эффективный метод формирования библиографических баз, имеющий свои преимущества и недостатки. К преимуществам нужно отнести автоматическое формирование описаний источников, гарантирующее отсутствие опечаток и ошибок. Отсюда же следуют и недостатки: алгоритмы автоматических процессов не способны охватить тонкости синтаксиса баз BibTeX, поэтому записи, полученные из интернета, часто требуют доработки. Разбирая синтаксис баз BibTeX, выделим моменты, на которые следует обратить внимание с этой точки зрения.

Библиографические базы представляют собой обычные текстовые файлы, которые можно править в любом редакторе, однако использование специализированной программы **Jabref** оградит вас от неосторожного удаления служебных символов, которое способно испортить не только отдельные записи, но и базу в целом.

Файлы библиографических баз, обычно имеющие расширение **•.bib**, содержат описания источников в виде записей специального формата, показанного в следующем примере:

⁴ Удобный интерфейс поиска публикаций предоставляет сайт www.doi.org


```

@BOOK(Kotelnikov-LaTeX-2004,
  author = {Котельников, И. and Чеботаев, П.},
  title = "Настольная издательская система
    {\upshape\LaTeXe} по-русски",
  address = {Новосибирск},
  publisher = {Сибирский хронограф},
  year = 2004,
  pages = 491,
  isbn = {978-5-7931-0878-9},
  language = "russian")

```

Программа `Jabref` предоставляет удобный интерфейс для просмотра и редактирования отдельных полей, а вывод записи «целиком» в виде, показанном выше, обеспечивает меню «`{}` BibTeX source».

В зависимости от описываемого источника записи делятся на типы. Тип задается словом, начинающимся символом `@`, в котором заглавные и строчные буквы не различаются, например, типы `@ARTICLE` и `@article` одинаковы.

Перечислим стандартные типы записей.

`@article` — статья в периодическом издании.

`@book`, `@inbook` — книга и ее часть с указанным диапазоном страниц, номеров глав и т. д.

`@incollection` — часть книги, имеющая отдельное название.

`@booklet` — брошюра без указания издателя.

`@conference`, `@proceedings` — сборник трудов конференции.

`@inproceedings` — статья в сборнике трудов конференции.

`@manual` — техническая документация.

`@masterthesis`, `@phdthesis` — дипломная работа и диссертация.

@techreport — опубликованный отчет.

@unpublished — неопубликованный документ, имеющий автора и заглавие, например, статья в электронном архиве.

@misc — документ, не попадающий ни в одну из перечисленных категорий.

Запись содержит описание источника, которое заключается в фигурные или круглые скобки. Описание начинается с ярлыка, служащего для идентификации источника и его цитирования командами `\cite{•}`. Ярлык должен быть уникальным, легко запоминаемым и хорошо распознаваемым. Обычно автоматически генерируемые ярлыки содержат информацию, ценную для издателей или составителей баз, но не всегда полезную для авторов. Такие ярлыки неудобны в работе и их следует заменить в соответствии с вашими собственными предпочтениями. Как показывает опыт, оптимальным набором для ярлыка служит фамилия первого автора, аббревиатура издания и год опубликования работы, например, `Ivanov-JETPL-2010`. Такой ярлык вы сразу вспомните, заглянув в публикацию.

За ярлыком следуют поля записи, разделяемые запятыми. Каждое поле состоит из названия и текста, разделенных знаком равенства. Текст заключается в двойные кавычки или фигурные скобки, которые можно опустить, если он содержит только цифры. Так как текст записей служит основой формирования окружения `thebibliography`, он может содержать строчные формулы и команды форматирования ЛАТ_EX.

Поля делятся на обязательные, дополнительные и необязательные.

Набор обязательных полей составляет информацию необходимую для поиска публикации, зависящую от ее вида. Например, описание книги должно содержать её название, данные об издательстве и годе выпуска, а для статьи, нужно указать название, список авторов, название журнала и год его выпуска.

Дополнительные поля хранят информацию, упрощающую поиск. При отсутствии обязательного поля BibTeX выдаст сообщение об ошибке, а при отсутствии дополнительного лишь предупреждение.

Необязательные поля BibTeX игнорирует. В них можно хранить комментарии и другую информацию, существенную с точки зрения автора. Записи могут содержать любые поля с нестандартными названиями, трактуемые как необязательные.

Перечислим названия и назначение стандартных полей.

address — адрес издательства или организации, опубликовавшей источник.

author, editor — список авторов или редакторов публикации.

booktitle — название книги или трудов конференции, содержащих публикацию.

chapter — номер главы, части или раздела книги.

doi — идентификатор публикации в сети Интернет.

eid — уникальный номер статьи в журнале, заменяющий номера страниц.

eprint — идентификатор публикации в сети или электронном архиве.

howpublished — поле, предназначенное для документов, способ издания которых не вписывается на в какую другую категорию.

institution — организация, опубликовавшая отчет.

journal — название журнала.

key — слово, используемое при сортировке, если не указаны поля **author** или **editor**.

language — язык документа. Данное поле используется стилями **gost2008**. Для правильной обработки документов на русском языке оно должно иметь значение **russian**.

month, year — месяц и год опубликования или написания документа.

note — дополнительная информация, которая выводится вместе с выходными данными публикации.

number, volume — номер и том издания.

organization — организация, опубликовавшая труды конференции или техническую документацию.

pages — номер первой страницы, или диапазон страниц публикации.

school — учебное заведение, в котором написана диссертация или дипломная работа.

series — название серии книг или трудов.

title — название публикации.

type — уточнение некоторых параметров публикации.

url — адрес доступа к публикации в сети Интернет.

В табл. 5.1 приведены стандартные типы записей и указаны их обязательные и дополнительные поля. Разберем особенности синтаксиса некоторых из них.

Поле **author** содержит список авторов публикации, которые разделяются ключевым словом **and**, например,

```
author = {Karl Marx and Frederick Engels}.
```

Если список авторов содержит обычное слово **and**, его следует заключить в фигурные скобки, например,

```
author = "Procter {and} Gambel".
```

Стандартные типы и поля библиографических записей⁵

Записи	author	title	journal	volume	number	year	month	pages	note	editor	publisher	address	isbn	edition	series	chapter	booktitle	organization	howpublished	school	institution	type	key	doi	url	eprint
@article	●	●	●	○	○	●	○	○	○														○	○	○	○
@book	*	●		.	.	●	○		○	*	●	○	○	○	○								○	○	○	○
@inbook	*	●		.	.	●	○	*	○	*	●	○	○	○	○	*						○	○	○	○	○
@incollection	●	●		.	.	●	○	○	○	○	●	○	○	○	○	○	●					○	○	○	○	○
@conference	●			.	.	●	○		○	○	○	○	○	○	○			○					○	○	○	○
@proceedings	●	●		.	.	●	○		○	○	○	○	○	○	○			○					○	○	○	○
@inproceedings	●	●		.	.	●	○	○	○	○	○	○	○	○	○		●	○					○	○	○	○
@booklet	○	●				○	○		○			○	○						○				○	○	○	○
@manual	○	●				○	○		○			○		○				○					○		○	○
@masterthesis	●	●				●	○		○			○								●		○	○	○	○	○
@phdthesis	●	●				●	○		○			○								●		○	○	○	○	○
@techreport	●	●			○	●	○		○			○									●	○	○	○	○	○
@unpublished	●	●				○	○		●														○	○	○	○
@misc	○	○			○	○	○		○										○				○	○	○	○

● обязательное поле, * должно присутствовать одно из полей, * должны присутствовать
 ○ дополнительное поле, . может присутствовать одно из полей, ○ одно или оба поля.

⁵ Все записи могут содержать дополнительное поле language, записи @article и @misc — дополнительное поле issn. Записи @article также могут содержать дополнительное поле eid.

Как имя, так и фамилия могут состоять из несколько слов: Johann Sebastian Bach, Johannes Chrysostomus Wolfgangus Theophilus (Gottlieb) Mozart, Ludwig von Beethoven. Когда требуется сократить имя до инициалов, очевидно, фамилия не должна сокращаться, поэтому BibTeX использует следующий алгоритм ее определения. Если последовательность слов не содержит запятой, то фамилией является последнее слово. Фамилия, состоящая из нескольких слов, должна стоять перед именами и отделяться от них запятой: von Beethoven, Ludwig.

Если в фамилии автора есть акцентированные буквы, а в скачанной записи диактрические знаки отсутствуют, их можно поставить с помощью команд, представленных в табл. 2.8, например, Antonin Dvo\v{r}\ak \rightsquigarrow Antonin Dvořák.

В библиографии принято указывать сокращенные названия журналов, а полное используется только, если сокращенное неизвестно. В то же время, в автоматически генерируемых записях поле `journal` обычно содержит полное название. Стандартное сокращение, если оно не указано на сайте журнала, можно найти в библиографиях цитируемых работ, а чтобы получить его в программе JabRef, достаточно нажать на иконку справа от поля `journal`.

Для названий журналов можно использовать аббревиатуры, снижающие вероятность опечаток. Аббревиатуры создают специальные записи

```
@STRING{ аббревиатура = "текст" }.
```

Например, `@string{prb = "Phys. Rev.~B"}`, определяет аббревиатуру `prb`, для журнала Physical Review B. Используются аббревиатуры следующим образом: `journal = prb`. В отличие от текста они не заключаются в скобки или кавычки. Обработывая запись, BibTeX заменит аббревиатуру заданным текстом. Определения аббревиатур должны предшествовать их использованию, поэтому записи `@string` следует размещать в начале библиографической базы. Для них также можно создать специальную базу и загрузить ее первой.

Название публикации может содержать формулы химических соединений, простые математические формулы, собственные имена и другие слова, содержащие заглавные буквы. Такие данные нужно внимательно отслеживать в автоматически генерируемых записях, так как часто они сделаны неправильно.

Поле `title` следует внимательно просмотреть и при необходимости отредактировать. Некоторые стили BibTeX формируют из него предложение, которое начинается с заглавной буквы, а остальные буквы переводятся в строчные. Фигурные скобки помогают защитить текст от подобных преобразований. В них следует заключить химические формулы, собственные имена, географические названия и т. д. Например, `{CeRu$_2$}`; не ходите дети в {Африку} гулять; распределение {Бозе-Эйнштейна}. Без защиты соединение CeRu_2 будет преобразовано в `ceru2` название континента и фамилии тоже потеряют заглавные буквы.

Текст, заключенный в фигурные скобки, может находиться в любом поле. Он не преобразуется и обрабатывается «как целое». При алфавитной сортировке записей фамилий блок `{von Beethoven}` окажется в группе ‘v’, а фамилия `von Beethoven` попадет в группу ‘B’, так как приставку `von` BibTeX распознает как титул.

Поле `type` используется для уточнения или корректировки параметров ссылки. В записях `@techreport` в нем можно указать тип отчета, например, `type = "Отчет по гранту РФ"` В записях `@phdthesis` уточнить вид диссертации,

```
type = "{Ph.~D} dissertation".
```

а в записях `@inbook` и `@incollection` с его помощью корректируют название раздела. Например, поле `chapter = "1.2"` генерирует номер главы \rightsquigarrow `chapter 1.2`, а уточнение `type = "section"` даст номер раздела \rightsquigarrow `section 1.2`.

В библиографической базе можно использовать перекрестные ссылки. Для этого определено специальное поле

```
crossref = {ярылык},
```

которое ссылается на ярлык другой записи той же базы. Перекрестное цитирование предназначено для описания ряда материалов, содержащихся в одном источнике. Например, поместив в запись `@proceedings{SampleConf, ...}` выходные данные трудов конференции, их можно не дублировать в записях, `@inproceedings`, содержащих данные о докладах. В них достаточно указать авторов, названия докладов, диапазоны страниц и поле `crossref = {SampleConf}`, тогда недостающая информация будет добавлена из данной записи. Записи с ссылками `crossref`, должны находиться в базе *do* записи, на которую они ссылаются. Отметим, что BibTeX автоматически добавит источник перекрестных ссылок в библиографию, если на него ссылаются более двух раз.

Закачав и отредактировав запись, полезно проверить ее, путем компиляции со стилем `plain` или `unsrt`. Данные стили наиболее чувствительны к ошибкам синтаксиса записей, так как преобразуют заглавные буквы в строчные в названиях источников. Как правило, они также позволяют «отловить» недопустимые unicode-символы (см. в конце разд. 2.7), часто встречающиеся в автоматически генерируемых записях.

Мы разобрали только основные особенности синтаксиса библиографических баз. Более полно они описаны в книгах [3, 4].

Часть II

Дополнительные
ВОЗМОЖНОСТИ

Глава 6

Стандартные операции

Л^AT_EX имеет обширный набор средств, хотя и являющихся стандартными, однако практически не используемых в оформлении публикаций. Их описанию посвящена данная глава. В ней обсуждаются приемы манипулирования боксами и счетчиками, создание и редактирование команд, а также тонкости использования пружин.

6.1. Создание и настройка команд

При верстке отчетов, диссертаций, дипломных работ и т. д., особенно написанных по-русски, зачастую возникает необходимость изменить некоторые команды, специально предназначенные для таких действий. В данном разделе обсуждаются средства, позволяющие это сделать, однако подавляющее большинство команд Л^AT_EX вовсе не предназначено для внешнего вмешательства. Их лучше не трогать или делать это в крайнем случае, с большой осторожностью, изучив сначала основы программирования T_EX и Л^AT_EX [1–4] и описание ядра Л^AT_EX [20].

Средства создания и изменения команд предназначены в основном для документов «личного пользования», так как издатели могут заблокировать такую возможность.

Авторы могут переопределять уже существующие команды и вводить новые, используя декларации

```
\newcommand{команда}[n][параметр]{определение} ,  
\renewcommand{команда}[n][параметр]{определение} ,  
\providecommand{команда}[n][параметр]{определение} .
```

Первая из них создает новую команду, вторая переопределяет уже существующую, а третья вводит новую команду, если она еще не определена, в противном же случае будет действовать существующая. Все они имеют вариант имени со звездочкой:

```
\newcommand*{команда}[n][параметр]{определение} ,  
\renewcommand*{команда}[n][параметр]{определение} ,  
\providecommand*{команда}[n][параметр]{определение} .
```

По умолчанию аргументы создаваемых команд могут содержать несколько абзацев текста, а чтобы ограничить их одним абзацем, команды нужно вводить декларациями «со звездочкой». Использовать новую команду можно только после ее определения.

Область действия деклараций ограничена группой, поэтому команды, созданные или измененные внутри группы, вне нее не действуют. Новые команды (изменения) следует вводить в преамбуле программы, тогда областью их действия будет весь документ.

Перечисленные декларации имеют одинаковый синтаксис. Разберем его на примере `\newcommand`. Первым аргументом является новая команда, а вторым — часть кода, который подставляется в программу при каждом ее выполнении. По умолчанию новая команда не имеет ни аргументов, ни параметров, но при необходимости ей можно присвоить до девяти аргументов, количество которых объявляется в первом параметре декларации `\newcommand`. В описании кода, выполняемого командой, аргументы обозначаются номерами `#1`, `#2`... Во втором параметре декларации `\newcommand` для аргумента `#1` можно задать значение по умолчанию, тогда он станет параметром новой команды. Создаваемая команда может иметь только один параметр.

Поясним сказанное примерами. Часто используемый пробел шириной `1em`, задаваемый командой `\quad`, оказавшись в начале или конце строки, игнорируется компилятором, поэтому он работает не всегда:

```
\quad \rule{1ex}{1ex} \hfill \rule{1ex}{1ex} \quad \\\
```

В данном примере квадратики, рисуемые командой `\rule`, заключены между пробелами и жесткой пружинкой `\hfill`, расталкивающей по краям строки.

Введем новую команду

```
\newcommand\Quad{\hspace*{1em}}
```

Она содержит команду `\hspace*{1em}`, которую компилятор обязан выполнять всегда, поэтому пробел `\Quad` появится, где бы он ни стоял:

```
\Quad \rule{1ex}{1ex} \hfill \rule{1ex}{1ex} \Quad \\\
```

Видно, что теперь квадратики сдвинулись от краев строки на ширину пробела.

Добавим возможность изменять ее, задав параметр:

```
\renewcommand\Quad[1][1em]{\hspace*{#1}}.
```

При использовании команды без параметра ширина пробела останется прежней:

```
\Quad \rule{1ex}{1ex} \hfill \rule{1ex}{1ex} \Quad \\\
```

А теперь, воспользовавшись параметром, утроим ее:

```
\Quad[3em]\rule{1ex}{1ex}\hfill\rule{1ex}{1ex}\Quad[3em]\\\
```

Как уже говорилось, наборы команд, используемых в тексте и формулах, различаются. Если команда, предназначенная для верстки текста, содержит математические символы, их нужно ввести с помощью команды

```
\ensuremath{математические команды}.
```

Для примера создадим команду `\TO`, связывающую в наших примерах код с результатом его выполнения. Она содержит символ `\leadsto` (\leadsto):

```
\newcommand\TO{\ensuremath{\leadsto}}
```

Теперь можно написать `\TO $\sin x \leadsto \sin x`. Если определить команду в виде строчной формулы, `\newcommand\TO{\\leadsto}`, ее нельзя будет использовать в других формулах. Наш же вариант команды работает везде: `$(a\TO b)$ \leadsto (a \leadsto b)`.

Следующий пример связан с изменением стандартных заголовков, которые выводят команды, собранные в табл. 6.1.

Таблица 6.1
Стандартные заголовки

Команды	Стандартные заголовки	
<code>\abstractname</code>	Abstract	Аннотация
<code>\appendixname</code>	Appendix	Приложение
<code>\bibname</code>	Bibliography	Литература
<code>\refname</code>	References	Список литературы
<code>\contentsname</code>	Contents	Оглавление
<code>\figurename</code>	Figure	Рис.
<code>\listfigurename</code>	List of Figures	Список иллюстраций
<code>\tablename</code>	Table	Таблица
<code>\listtablename</code>	List of Tables	Список таблиц
<code>\indexname</code>	Index	Предметный указатель
<code>\chaptername</code>	Chapter	Глава
<code>\partname</code>	Part	Часть

Заголовок `\bibname` используют классы `book.cls` и `report.cls`, а `\refname` — `article.cls`.

Заданное по умолчанию значение изменяется с помощью переопределения `\renewcommand`. Заменим, например, *Оглавление* на *Содержание*:

```
\renewcommand\contentsname{Содержание}
```

Аналогично редактируются все остальные заголовки, при этом можно также настроить их вывод. Например, заголовок *Приложение*, измененный следующим образом:

```
\renewcommand\appendixname{%  
\vspace*{-2\baselineskip}\mbox{\hfill Приложение},
```

сместится вверх и прижмется к правому краю страницы.

Пример создания команды с параметром приведен на с. 207.

Мы разобрали пользовательский уровень настройки команд ЛАТ_EX. На системном уровне для этих целей привлекают средства Т_EX [1]. В ряде случаев они оказываются необходимы, поэтому рассмотрим вкратце и их.

Для определения команд используются команды¹

```
\def команда # {определение} ,  
\gdef команда # {определение} ,
```

где команда — имя новой или переопределяемой команды, # — список аргументов, используемых в определении. Имя команды набирается без скобок. Список может отсутствовать или содержать до девяти аргументов, перечисляемых без скобок и запятых, например #1#2 и т. д.

Существенным отличием команд `\def` и `\gdef` является отсутствие механизма контроля ЛАТ_EX. Каждая из них заменяет собой `\newcommand` и `\renewcommand`. Если вводимая команда уже имеется, она переопределяется без предупреждений и сообщений об ошибке. Действие команды, определяемой `\def`, ограничено группой, в которой она находится, а команда, вводимая `\gdef`, действует глобально, ее область действия ничем не ограничена.

Помимо определения команд имеется возможность присваивания одной команде действие другой с помощью конструкции

```
\let\B=\A ,
```

в которой знак равенства очень часто опускают. На момент присваивания команда `\B` может быть не определена, но `\A` должна иметь значение. Если команда `\A` имеет аргументы и параметры, `\B` автоматически наследует и их. Если впоследствии команда `\A` изменяется, `\B` остается неизменной.

¹ Команды `\def` и `\gdef` имеют более гибкий синтаксис, для анализа которого нужны глубокие знания Т_EX. Здесь описан его простейший вариант.

В качестве примера введем короткую команду, печатающую обратный слэш, `\let\bs\textbackslash`, и сразу же воспользуемся ей: `\bs ~ \`.

Имена команд более низкого уровня, чем макрокоманды, с которыми имеют дело авторы, часто содержат символ `@`, служащий защитой от их случайного изменения. Чтобы редактировать такие команды, нужно сначала с помощью декларации

```
\makeatletter
```

дать символу `@` статус буквы, а внося изменения, вновь вернуть статус символа декларацией

```
\makeatother .
```

Поясним сказанное важным примером. В стандартных списках литературы номера источников заключаются в квадратные скобки, а в традициях российской полиграфии они отделяются от описаний источников точкой. Выводит номер команда

```
\@biblabel{номер или метка} .
```

Ее редактирование напрямую недоступно, но применив конструкцию:

```
\makeatletter
\def\@biblabel #1 {#1.}
\makeatother,
```

получим команду `\@biblabel{•}`, которая будет выводить свой аргумент, завершая его точкой.

Заключительное замечание касается правильной организации области действия новых команд. Создадим команду

```
\newcommand\cmd[1]{\texttt{\textbackslash#1}},
```

предназначенную для печати команд.² Ее аргументом является имя команды: `\cmd{sample} ~ \sample`.

Было бы ошибкой сократить код с помощью декларации `\tt:`

```
\newcommand\cmd[1]{\tt\textbackslash#1},
```

так как в этом случае изменение шрифта затронет весь последующий текст. Чтобы не ограничить действие деклараций новой

² Обычно для этого используют стандартную команду `\verb` (см. с. 64), но она не работает в аргументах других команд, что не очень удобно.

команды, группа для нее не создается. Автор сам должен отрегулировать группирование дополнительными скобками в ее определении, например:

```
\newcommand\cmd[1]{\tt\textbackslash#1}.
```

Теперь шрифтом `\tt` будет печататься только имя команды.

Мы рассмотрели далеко не все тонкости определения новых команд. За рамками обсуждения оказались также средства создания и переопределения окружений. Но этого краткого введения достаточно для настройки большинства команд, с которыми сталкиваются авторы. За более подробной информацией заинтересованный читатель может обратиться к книгам [1–4].

6.2. Новые окружения

Новые окружения вводят декларации

```
\newenvironment{имя}[число][значение]{перед}{после} ,  
\newenvironment*{имя}[число][значение]{начало}{конец} ,
```

а существующие переопределяют декларации

```
\renewenvironment{имя}[число][значение]{начало}{конец} ,  
\renewenvironment*{имя}[число][значение]{начало}{конец} .
```

Хотя последние декларации позволяют изменять существующие окружения, редактировать стандартные окружения крайне опасно, так как последствия могут оказаться непредсказуемыми.

Тело создаваемых окружений может содержать несколько абзацев текста, но если их имя содержит звездочку, оно ограничено одним абзацем.

Синтаксис всех перечисленных деклараций одинаков. Их первым аргументом является имя окружения. В первом параметре указывается число аргументов окружения, которое не должно быть больше девяти. Окружение может также иметь один параметр, в качестве которого берется первый аргумент. В этом случае его значение, используемое по умолчанию, заносится во второй параметр деклараций.

Аргументы окружения могут использовать команды, находящиеся в аргументе `перед`.³ При выполнении программы они замещают команду `\begin{имя окружения}` и выполняются перед обработкой тела окружения. Их действие, включая действие вновь определяемых команд, распространяется на всю остальную часть окружения.

Команды, выполняемые после обработки тела окружения, помещаются в последний аргумент. При выполнении программы они замещают команду `\end{имя окружения}`. Эти команды могут иметь собственные аргументы, однако аргументы окружения им недоступны.

В аргументах `перед` и `после` могут вводиться новые команды. При выходе из окружения они становятся неопределенными.

В качестве примера создадим окружение `mytemize` и команды `\itemlabels` и `\itemlabelset`, которые будут формировать настраиваемый список `itemize`:

```
\newcommand\itemlabelset[4]{
  \renewcommand\labelitemi{\labelitemfont #1}
  \renewcommand\labelitemii{\labelitemfont #2}
  \renewcommand\labelitemiii{\labelitemfont #3}
  \renewcommand\labelitemiv{\labelitemfont #4} }
\newcommand\itemlabels{
  \itemlabelset{\textbullet}{\textasteriskcentered}
                {\ensuremath\circ}{\textperiodcentered} }
\itemlabels
\newenvironment{mytemize}[1][\itemlabels]
  {#1 \begin{itemize}} {\end{itemize}}
```

Команда `\itemlabelset` переопределяет стандартные команды `\labelitem*`, печатающие метки записей четырех уровней списков. Ее аргументами являются символы меток. Шрифт меток `\labelitemfont` обычно устанавливается в классе документа как `\normalfont`.

³ Вместо команд в двух последних аргументах может находиться обычный текст, выводимый `перед` или `после` тела окружения.

Команда `\itemlabelset` оказывается очень громоздкой, так как в качестве аргументов, чтобы не возникли проблемы с кодировками, нужно использовать не сами символы, а хранящие их команды. Поэтому для удобства использования `\itemlabelset` добавлена команда `\itemlabels`, формирующая определенный набор меток. Созданный набор из черного кружочка, звездочки, белого кружочка и центрированной точки будет использоваться по умолчанию.

Новое окружение `mytemize` имеет параметр, по умолчанию выполняющий команду `\itemlabels`. Непосредственно перед текстом окружения выводится команда `\begin{itemize}`, а после него — `\end{itemize}`, поэтому текст является списком.

В параметре окружения `mytemize` можно указать команды настройки списков, описанные в разд. 9.7. Изменив верстку формируемого списка, они не затронут остальные.

Воспользуемся новым окружением и сделаем два списка, поместив их в министраницы (см. с. 52), чтобы вывести на одном уровне:

- Первый уровень. ■ Первый уровень.
- * Второй уровень. ■ Второй уровень.
- Третий уровень. ■ Третий уровень.
- Четвертый уровень. ■ Четвертый уровень.

Левый список сверстала следующая совокупность команд:

```
\begin{mytemize}[\setlength\leftmargini{11pt}]
\item Первый уровень.
\begin{mytemize}
\item Второй уровень...
\end{mytemize}
```

Команда `\setlength` в параметре первого окружения корректирует левую отбивку списка так, чтобы записи первого уровня выровнялись по левой границе страницы.⁴ Остальные окружения параметров не имеют.

⁴ Подробнее о настройках списков см. разд. 9.7.

Окружения второго списка также использованы без параметров. Их метки в форме квадратиков, нарисованных командой `\rule`, сформировала переопределенная перед созданием списка команда

```
\renewcommand\itemlabels{
  \itemlabelset {\rule[.05ex]{1ex}{1ex}}
    {\rule[.1ex]{.9ex}{.9ex}} {\rule[.2ex]{.8ex}{.8ex}}
      {\rule[.3ex]{.7ex}{.7ex}} }.
```

Когда кружение `mytemize` используется без параметра, новый набор меток актуализируется автоматическим выполнением команды `\itemlabels`, но если в параметр добавлены настройки верстки списка, команда `\itemlabels` не выполняется. Тогда ее можно добавить к настройкам, или выполнить перед созданием списка.

Следует подчеркнуть, что команды `\itemlabelset` заменяет набор стандартных меток окружения `itemize` нестандартными, и чтобы вернуться к стандартному набору, нужно выполнить ее со следующими аргументами

```
\itemlabelset{\textbullet}{\bfseries \textendash}
  {\textasteriskcentered}{\textperiodcentered}
```

6.3. Операции с боксами

Боксы, представляющие собой прямоугольники определенной ширины, высоты и глубины, — основные элементы верстки. Они могут содержать отдельные символы, несколько абзацев текста, формулы, рисунки, таблицы, а также другие боксы. Пожалуй, единственным ограничением является то, что бокс не может разбиваться на страницы.

В предыдущей части уже обсуждалось создание простейших боксов командами `\mbox` и формирование министраниц со сложной структурой с помощью окружений `minipage`. Теперь рассмотрим операции, предназначенные для манипулирования боксами.

Зафиксировать ширину бокса-строки позволяет команда

```
\makebox[ширина] [выравнивание] {текст}.
```

Без параметров она эквивалентна команде `\mbox`, создающей бокс с шириной равной длине текста. Если в параметре `\makebox` указана ширина, компилятор использует ее при встраивании бокса в строку вне зависимости от того, какую длину имеет текст. Если объявленная ширина окажется меньше его длины, он наложится на окружающий текст.

Второй параметр команды `\makebox` задает метод выравнивания текста, относительно границ бокса:

- `c` — центрирование,
- `l` — выравнивание слева,
- `r` — выравнивание справа,
- `s` — выравнивание с обеих сторон.

Приведем примеры, наглядно показывающие методы выравнивания боксов `\makebox` и их взаимодействие с окружающим текстом. В примерах границы окружающего текста представлены вертикальными линиями, в левом и правом столбцах показаны боксы с коротким и длинным текстом, а в центральном — формирующие их команды:

a a a	\makebox[4em] [c] {•}	длинный текст
a a a	\makebox[4em] [l] {•}	длинный текст
a a a	\makebox[4em] [r] {•}	длинный текст
a a a	\makebox[4em] [s] {•}	длинный текст

В параметре команды `\makebox` с помощью длин

```
\width, \height, \depth, \totalheight
```

доступны значения ширины, высоты, глубины и полной высоты бокса, равной сумме высоты и глубины. Сравните

$$\fbox{i} \rightsquigarrow \boxed{i} \quad \text{и} \quad \fbox{\makebox[\height]{i}} \rightsquigarrow \boxed{i}.$$

Бокс можно заключить в рамку. Для этого предназначены команды

```
\fbox{текст},
```

`\framebox[ширина][выравнивание]{текст}`, действующие аналогично `\mbox` и `\makebox`. Длины `\fboxrule` и `\fboxsep` хранят толщину рамки и ширину поля, отделяющего ее от текста.

Бокс можно сдвигать вверх и вниз с помощью команды

`\raisebox{сдвиг}[высота][глубина]{текст}`.

Ее первый аргумент задает величину сдвига. Положительная длина соответствует сдвигу вверх, а отрицательная — вниз. Так как данная операция раздвигает соседние строки, параметры позволяют скорректировать вертикальные размеры бокса. При встраивании в строку объявленные `высота` и `глубина` используются вместо истинных размеров бокса. Например, команда

`\raisebox{-0.5ex}{\height}{\depth}{x}` \rightsquigarrow x ,

сдвигая букву вниз, полностью маскирует сдвиг. Здесь мы воспользовались оригинальными размерами бокса, которые доступны в параметрах.

Пакет `graphicx` вводит ряд команд, позволяющих масштабировать и вращать боксы. Среди них команда

`\scalebox{масштаб}[вертикальный масштаб]{текст}`

масштабирует бокс относительно его собственных размеров. Ее первым аргументом является масштаб в виде действительно-го числа, например `\scalebox{1.25}{текст}` \rightsquigarrow ТЕКСТ. Число может быть отрицательным, тогда вместе с масштабированием производится еще и отражение:

`текст\scalebox{-1}{текст}` \rightsquigarrow текст_{ЛОЖАЛ}.

Если параметр отсутствует, бокс масштабируется пропорционально. Параметр позволяет установить разные коэффициенты горизонтального и вертикального масштабирования:

`\scalebox{1}[1.5]{текст}` \rightsquigarrow ТЕКСТ.

Производной от `\scalebox` является упоминавшаяся ранее команда зеркального отражения

`\reflectbox{текст} ≡ \scalebox{-1}[1]{текст}` ,

использованная в примере рисунка в конце разд. 3.3.

Подогнать бокс под заданные размеры позволяют команды

`\resizebox{ширина}{высота}{текст}` и

`\resizebox*{ширина}{полная высота}{текст}` .

Они отличаются тем, что в самой команде второй аргумент определяет высоту бокса относительно базисной линии строки, а в варианте со звездочкой — его полную высоту. Если при масштабировании необходимо сохранить пропорциональность, один из аргументов заменяется восклицательным знаком, например

`\resizebox{1em}!{n} ∼ n` .

Аналогично `\makebox` в аргументах данных команд доступны оригинальные размеры бокса. Воспользовавшись этим, изменим предыдущий пример:

`\resizebox{1em}\width{n} ∼ n` .

Вращение бокса обеспечивает команда

`\rotatebox[список параметров]{угол}{текст}` .

Положительный и отрицательные углы задают, соответственно, вращение либо по часовой стрелке, либо против часовой стрелки. Параметры, перечисляемые списком через запятую, позволяют настроить положение оси вращения и задать единицы измерения угла.

`origin` = координаты — положение оси вращения, заданное координатами, привязанными к границам бокса или базисной линии строки. Горизонтальная координата может принимать значения `l`, `r` и `c`, что соответствует левой, правой границе и середине ширины бокса. Вертикальная координата `t`, `b` и `s` привязывается к верхней, нижней границе и середине высоты бокса, или положению базисной линии строки `B`. По умолчанию ось вращения находится на левой границе бокса в точке его привязки к базовой линии, `origin=lB`, а значение `origin=c` соответствует вращению относительно его центра.

x = длина — смещение оси вращения, от левой границы бокса.

y = длина — смещение оси вращения, от нижней границы бокса.

$units$ = число — единицы угла поворота. Число задает полный оборот, поэтому $units=6.28319$ соответствует отсчету угла в радианах.

Для сравнения приведем несколько примеров, явно показав в них базовую линию:

```
\rotatebox{180}{пример}           ~> — дэвидп — ,  
\rotatebox[origin=c]{180}{пример} ~> — дэвидп — ,  
\rotatebox[x=.15\width]{-10}{пример} ~> — пример — .
```

Как видно из последнего примера, смещение оси вращения можно задавать, используя оригинальные размеры бокса, которые определены в параметре команды `\rotatebox`.

Бокс можно сохранить, чтобы многократно использовать впоследствии. Для этого прежде всего с помощью объявления

```
\newsavebox{команда} .
```

нужно ввести команду, которая будет его хранить.

Записать в нее бокс позволяют команды

```
\sbox{команда}{текст} ,
```

```
\savebox{команда}[ширина][выравнивание]{текст} .
```

Они формируют бокс полностью аналогично тому, как это делают команды `\mbox` и `\makebox`, но не вносят его в документ. Для сохранения бокса определено и также окружение

```
\begin{lrtext}{команда}
```

строка текста

```
\end{lrtext}
```

отличающееся от команд тем, что пробелы, находящиеся в начале и конце строки текста, игнорируются. Кроме этого в нем можно использовать команду `\verb` и окружение `verbatim`.

Сохраненный бокс выводит команда

```
\usebox{команда}.
```

Приведем пример сохранения и использования бокса:

```
\newsavebox\Tiger
\sbbox\Tiger{\includegraphics{tiger}}
\newcommand\tiger[1][.1]{\scalebox{#1}{\usebox\Tiger}}
\newcommand\regit[1][.1]{\scalebox{-#1}[#1]{\usebox\Tiger}}
\mbox{\hfil \regit\ \tiger}
```



Создав команду `\Tiger`, мы сохранили в ней бокс, загрузив в него изображение тигра. Затем ввели две новых команды `\tiger[•]` и `\regit[•]`, использующие этот бокс и масштабирующие картинку, при этом `\regit` еще и отражает ее. По умолчанию картинка уменьшается в десять раз, но параметр команд позволяет задать любой другой размер. В конце мы вывели обе картинки, отцентрировав их с помощью пружины, и вставили между ними пробел, чтобы тигры не щекотали друг друга усами.

6.4. Операции со счетчиками

Авторам дано право вводить новые счетчики, а также изменять подчиненность и формат вывода существующих.

Новый счетчик вводится командой

```
\newcounter{счетчик}[внешний счетчик].
```

Если внешний счетчик не указан, созданный счетчик будет иметь сквозную нумерацию, как у страниц и примечаний. Если указан внешний счетчик, новый счетчик будет для него подчиненным,

при этом сам внешний счетчик может быть в подчинении у другого счетчика. Например, счетчик `subsection`, являясь внешним для `subsubsection`, подчинен счетчику `section`. Подчиненность счетчиков передается по цепочке до самого верха, а их значения могут сбрасываться при операциях с внешними счетчиками. Счетчик `subsubsection` сбрасывается при создании раздела `\subsection` и выше, так как является подчиненным для `subsection`, `section` и т. д. Изменить подчиненность счетчика помогает декларация

```
\counterwithin{счетчик}{внешний счетчик}.
```

Используя ее в преамбуле, можно установить нумерацию формул по разделам, `\counterwithin{equation}{section}`. Декларация

```
\counterwithout{счетчик}{внешний счетчик}
```

позволяет отменить подчиненность счетчика. Например, в классе `book` рисунки нумеруются по главам, а использование в преамбуле документа команды `\counterwithout{figure}{chapter}` установит их сквозную нумерацию.

Для изменения подчиненных счетчиков определена команда

```
\stepcounter{счетчик}.
```

Она увеличивает значение счетчика на единицу и сбрасывает в ноль подчиненные ему счетчики. Команда

```
\refstepcounter{счетчик},
```

изменяя значение счетчика, дает возможность связать его с меткой. Она вызывает `\stepcounter` для обновления счетчика и затем определяет системную команду `\@currentlabel`, в которую помещает его новое значение. Если после `\refstepcounter` поставить команду `\label`, метка присвоит себе значение счетчика.

Рассмотрим теперь настройку нумерации разделов. Хотя все они имеют собственные счетчики, начиная с какого-то уровня нумерацию можно отменить.

Уровни разделов имеют числовой идентификатор, принимающий значения -1 для `\part`, 0 для `\chapter`, 1 для `\section` и

т. д. до 5 для `\ subparagraph`. Установка счетчика `secnumdepth` в одно из этих значений ограничивает максимальный уровень нумерации разделов. Например, при его значении, равном двум, разделы `\ subsubsection` и ниже нумероваться не будут.

Еще один счетчик `tocdepth` ограничивает максимальный уровень разделов, вносимых в оглавление. Если установить его в единицу, в оглавление попадут разделы уровня `\ section` и выше.

По умолчанию класс `article` устанавливает значения счетчиков `secnumdepth` и `tocdepth` равными трем, а классы `book` и `report` — двум.

Как уже говорилось в разд. 1.4.6, текущее значение счетчика можно получить с помощью команды `\ theсчетчик`. Для независимых счетчиков выводится только одно число, равное их значению: `\ thepage` \rightsquigarrow 162. Перед значениями подчиненных счетчиков обычно выводятся значения внешних. Например, данный раздел имеет номер `\ thesection` \rightsquigarrow 6.4.

Для вывода счетчиков определено несколько форматов, представленных в табл. 6.2. Аргументом команд формата является имя счетчика. Наиболее часто используются арабские цифры, реже всего — символы из нижней ячейки левого столбца, обозначающие числа от 1 до 9. Остальные значения для данного формата не определены. Форматы с римскими цифрами не имеют нуля, но могут представить любое число, большее или равное единице. Форматы с латинскими и русскими буквами выводят числа от единицы до, соответственно, двадцати шести и двадцати восьми.

Таблица 6.2

Форматы вывода счетчиков

Форматы	Цифры	Форматы	Числа
<code>\ arabic{•}</code>	0 – 9	<code>\ Alph{•}</code>	A–Z
<code>\ Roman{•}</code>	C, I, L, M, V, X	<code>\ alph{•}</code>	a–z
<code>\ roman{•}</code>	c, i, l, m, v, x	<code>\ Asbuk{•}</code>	A–Я
<code>\ fnsymbol{•}</code>	*, †, ‡, §, ¶, , **, ††, ‡‡	<code>\ asbuk{•}</code>	a–я

Чтобы изменить формат вывода, например на большие римские цифры, нужно переопределить команду `\thesчетчик`:

```
\renewcommand{\thesчетчик}{\Roman{счетчик}}
```

```
\renewcommand{\thesчетчик}{\theвнешний.\Roman{счетчик}}
```

Первый вариант предназначен для независимого счетчика, а второй для подчиненного. Точку, разделяющую значения счетчиков, можно заменить любым другим символом. Например, по умолчанию значение счетчика рисунков отделяется от счетчика глав точкой и выводится цифрами: `\ref{f:DocPage}` \rightsquigarrow 1.2.

Переопределив вывод счетчика:

```
\renewcommand\thefigure{\thechapter-\Roman{figure}},
```

получим следующий результат: `\ref{f:DocPage}` \rightsquigarrow 1-II.

В разд. 1.4.6 говорилось, что с помощью команд `\setcounter` и `\addtocounter` значения счетчиков можно изменять. Стандартный пакет `calc` [21] коллекции `tools` позволяет проводить арифметические вычисления аргументов этих команд, используя операции сложения (+), вычитания (-), умножения (*) и деления (/).

В вычислениях операндами могут выступать числа и значения счетчиков, получаемые с помощью команды

```
\value{счетчик},
```

которая выводит целое число, равное значению счетчика. Команда `\value` не предназначена для печати значений счетчиков, ее использование в тексте вызовет ошибку. Значение

```
\value{page}  $\rightsquigarrow$  5
```

используется в дальнейших примерах.

Так как значения счетчиков целочисленны, основное правило вычислений сводится к тому, что результат каждой операции становится целочисленным. В операциях деления дробная часть отбрасывается без округления, поэтому выражения

```
\value{page}/6  $\rightsquigarrow$  5/6 и \value{page}/2  $\rightsquigarrow$  5/2
```

равны нулю и двум.

Вычисления проводятся слева направо. Круглые скобки используются для группирования выражений. В цепочке вычислений дробная часть отбрасывается при каждой операции, поэтому

выражение $2*\backslash\text{value}\{\text{page}\}/6*5 \rightsquigarrow 10/6*5 \rightsquigarrow 1*5$ равно пяти, а выражение $2*(\backslash\text{value}\{\text{page}\}/6)*5 \rightsquigarrow 2*0*5$ равно нулю, так как нулю равен результат деления в скобках.

Вещественные числа напрямую в вычислениях использовать нельзя. Они вводятся командой

```
\real{число}.
```

Их нельзя прибавлять или вычитать.

В операциях умножения и деления каждое вещественное число задается командой `\real`. Результат произведения или деления целого и вещественного числа становится целым, поэтому выражение

$\backslash\text{value}\{\text{page}\}*\backslash\text{real}\{1.5\}*\backslash\text{real}\{1.5\} \rightsquigarrow 5*1.5*1.5 \rightsquigarrow 7*1.5$
равно десяти.

Выражения не могут состоять только из вещественных чисел. Цепочка произведений и делений должна начинаться целым числом: $2*\backslash\text{real}\{1.5\} \dots$ — правильно, $\backslash\text{real}\{1.5\}*2 \dots$ — ошибка.

Пример вычисления счетчика приводится после обсуждения операций, производимых с длинами.

6.5. Операции с длинами

Во многих случаях пружины — простой и удобный инструмент верстки, поэтому рассмотрим их устройство и возможности настройки.

Пружины являются пробелами с размерами, задаваемыми длинами. Как говорилось в разд. 2.1, длины могут иметь постоянную величину, изменять свой размер в определенных пределах, а также расширяться до любого размера. Они определяются основным размером и параметрами растяжимости и сжимаемости, которые указываются после ключевых слов `plus` и `minus`, например `12pt plus 2pt minus 3pt`.

Новую длину вводит команда

```
\newlength{команда}.
```

С помощью команды `\setlength` ей нужно присвоить значение. Основной размер задается обязательно, а параметры можно опустить, тогда размер длины фиксируется. При наличии параметров длина становится упругой и может удлиняться на величину растяжимости или укорачиваться на величину сжимаемости. Основной размер и параметры могут быть положительны или отрицательны, но отрицательная растяжимость или сжимаемость вряд ли имеют практический смысл.

Доля длины, заданной командой, используемая в какой-либо другой команде, утрачивает свойство растяжимости и сжимаемости. Пробел, заданный командой `\hspace{2\length}` с длиной `\length`, равной `12pt plus 4pt`, будет иметь фиксированную ширину `24pt`.

Если основной размер равен нулю, например `0pt plus 4pt`, то длина изменяется от нуля до величины растяжимости. Пробел, заданный упругой длиной, подстраивая свой размер, не расталкивает объекты. Такая длина не может служить материалом для «изготовления» пружин, так как не имеет жесткости.

Величина длины определяется значением и размерностью. Для растяжимости пружин введена специальная размерность, которую мы назовем *жесткостью*. В \LaTeX определены три единицы жесткости:

`fil`, `fill`, `filll`.

При задании длины они указываются как обычные единицы. Чем больше букв «l» в размерности, тем больше жесткость. Команды `\hspace{0pt 1fil}` и `\hspace{0pt 1fill}` являются аналогами пружин `\hfil` и `\hfill`.

В виде команды \LaTeX вводит единственную длину `\fill`, обладающую жесткостью, равной `0pt plus 1fill`. Определена также команда, позволяющая пользоваться длиной с растяжимостью, заданной долями жесткости `fill`:

`\stretch{число}`,

аргументом которой является десятичное число. Например, пробел `\hspace{\stretch{2.5}}` — пружина с жесткостью `2.5fill`.

Пружины с одинаковым уровнем жесткости, т. е. одинаковой размерности, конкурируют между собой, а более жесткая сминает любую более мягкую. Покажем это на простом примере:

```
\hfill\hspace{0pt plus 2fill}\hspace{0pt plus 100fil} \\  
|                                     |                                     ||
```

Между первыми вертикальными палочками вставлена пружина `\hfill`, между второй и третьей — пружина жесткостью `2fill`. Жесткость пружины, стоящей между третьей и четвертой палочками, составляет `100fil`. Строка разорвана командой `\`, неявно вставляющей в конце пружину `\hfill`.

Из примера видно, что обе пружины с размерностью `fil` смяты, а расстояния между палочками пропорциональны жесткости пружин размерности `fill`.

Некоторые пакеты используют для выравнивания не мягкие, а жесткие пружины. Например после загрузки пакета `colortbl` (см. далее) ячейки таблиц выравниваются пружинами с жесткостью `fill`. В таких случаях для верстки может понадобиться более жесткая пружина, которую логично оформить в виде команды `\hfilll`. Это делается следующим образом:

```
\providecommand\hfilll{\hspace{0pt plus 1filll}}.
```

В определении использована команда `\providecommand`, так как может быть какой-то пакет уже позаботился ввести недостающую пружину, или сделает это в будущем.

Следует отметить, что все сказанное выше в равной степени применимо и к вертикальным пробелам, и к пружинам.

Обсудим теперь вычисление значений длин, устанавливаемых командами `\setlength` и `\addtolength` (см. разд. 2.1). Это позволяет делать уже упоминавшийся пакет `calc` [21]. Правила вычислений длин и счетчиков различаются. Значения длин задаются вещественными числами, которые в ходе вычислений не округляются.

Длины задаются явно или с помощью команд. В вычислениях могут участвовать длины разной размерности, например в выра-

жения `\somenlength -1mm` длина `\somenlength` может иметь любую размерность. Длины можно складывать и вычитать, умножать на число, а также делить на число или другую длину. В качестве чисел могут выступать значения счетчиков.

В операциях умножения с помощью оператора `*` длина должна быть первым операндом, поэтому выражение `1ex*3` корректно, а `3*1ex` — нет.

Ранее многократно использовался способ умножения, назовем его «умножением слева», в котором число стоит перед длиной, а знак `*` отсутствует. На примере упругих длин посмотрим, чем отличаются друг от друга два способа умножения:

```
\setlength\test{2pt plus 2pt minus 2pt}
\setlength\Length{2\test}                \the\Length ~ 4.0pt
\setlength\Length{\test*2}
\the\Length ~ 4.0pt plus 4.0pt minus 4.0pt
\setlength\Length{2\test +\test*2}
\the\Length ~ 8.0pt plus 4.0pt minus 4.0pt
\setlength\Length{2\test*2}              \the\Length ~ 8.0pt
```

Длина `\test` имеет растяжимость и сжимаемость. Длина `\Length` получается ее умножением на двойку разными способами. Видно, что умножение слева действует только на основной размер длины, при этом растяжимость и сжимаемость отбрасываются. Оператор `*` умножает каждую компоненту длины. В вычислениях можно комбинировать оба способа умножения.

В операциях умножения и деления на число длину, заданную явно, можно указывать без скобок:

```
2pt plus 1pt minus 2pt *2    ~ 4.0pt plus 2.0pt minus 4.0pt.
2pt plus 1pt minus 2pt /2   ~ 1.0pt plus 0.5pt minus 1.0pt.
```

Операторы `*` и `/` ставятся после последней ее компоненты. Операции с отдельными компонентами не поддерживаются, поэтому выражения `2pt*2 plus 1pt minus 2pt` и `2pt plus 1pt /2 minus 2pt` некорректны.

При умножении и делении на вещественное число растяжимость и сжимаемость упругой длины отбрасываются:

2pt plus 1pt minus 2pt * $\real{1.5}$	\rightsquigarrow	3.0pt.
2pt plus 1pt minus 2pt / $\real{1.5}$	\rightsquigarrow	1.33334pt.

С помощью команды

```
\ratio{длина}{длина}
```

две фиксированные длины можно поделить друг на друга. Результатом операции будет действительное число. Использование в числителе или знаменателе упругой длины даст ошибку.

Для примера вычислим, сколько четверостиший уместится на странице. Пусть в расчетах определены счетчик `verse`, а также длины `\numerator` и `\denominator`.

Сначала вычислим высоту одной строфы и вычтем ее из высоты текста на странице:

```
\setlength\numerator{\textheight -
\baselinestretch\baselineskip*4}.
```

Напомним, что высоту строки задает произведение коэффициента `\baselinestretch` на длину `\baselineskip`.

Теперь вычислим высоту одной строфы с учетом пустой строки, разделяющей строфы, и вертикального пробела `\parskip`, вставляемого между абзацами-строфами:

```
\setlength\denominator{\baselinestretch\baselineskip*5
+ \parskip}.
```

И наконец, получим число строф, разделив первую длину на вторую и добавив строфу, вычтенную из высоты текста:⁵

```
\setcounter{verse}{1*\ratio{1\numerator}{1\denominator} +1}
\theverse  $\rightsquigarrow$  7.
```

Последние действия учитывают специфику вычисления счетчиков и длин. Так как команда `\ratio` дает действительное число, ее результат умножается на единицу. Чтобы длины числителя и знаменателя заведомо были жесткими, они умножаются слева на единицу.

⁵ Последняя строфа на странице отличается от остальных тем, что у нее отсутствует вертикальная отбивка снизу.

Полученное значение показывает, что для поэмы из ста строф понадобится пятнадцать страниц. Пять строф остается в «запасе». На их месте можно разместить название поэмы и эпиграф.

Возможность использования в вычислениях параметров фрагмента текста, обеспечивают команды

```
\settowidth{команда}{текст} ,  
\settodepth{команда}{текст} .  
\settoheight{команда}{текст} ,  
\settototalheight{команда}{текст} ,
```

сохраняющие ширину, глубину, высоту и полную высоту текста в длины, которые должны быть заранее объявлены в виде команд. Сам текст не печатается. Три первые команды определены в ядре L^AT_EX, а последнюю добавляет пакет `calc`. Он также вводит ряд команд:

```
\widthof{текст} ,      \heightof{текст} ,  
\depthof{текст} ,    \totalheightof{текст} ,
```

выполняющих две функции. Они печатают текст и при этом являются длинами, равными соответствующим размерам. В печатном виде текст может насчитывать несколько строк, но при вычислении размеров он представляется в виде единой длинной строки. Когда перечисленные команды находятся в аргументах команд `\setlength` или `\addtolength`, текст не печатается.

Для примера присвоим длине `\Lgth` значение длины `\widthof` и напечатаем его:

```
\setlength\Lgth{\widthof{...}}  
\the\Lgth  $\rightsquigarrow$  404.88081pt      \the\linewidth  $\rightsquigarrow$  321.51613pt.  
Аргументом команды \widthof послужила предыдущая фраза, длина которой, как видно, превышает длину строки страницы.
```

Глава 7

Цветная верстка

Не принято раскрашивать текст научной публикации, но некоторые компоненты электронных документов можно выделять цветом. Например, пакет `hyperref` использует разноцветные рамки и цифры для выделения гиперссылок. Как правило, иллюстрации делаются цветными и, если красные кружки, синие квадраты и зеленые треугольники использованы при построении кривых, вполне допустимо в подписи к рисунку указать: «Символами ●, ■ и ▲ помечены кривые. . . ».¹

Возможность цветной верстки предоставляет стандартный пакет `color` [22] из коллекции `latex-graphics` [13]. Он имеет следующие настройки:

`dvipsnames` — обеспечивает загрузку 68 предустановленных цветов, представленных на рис. 7.1;









`usenames` — позволяет использовать названия этих цветов в командах цветной верстки;

`monochrome` — отключает действие команд цветной верстки, позволяя оставить их в программе.

¹ При печати на монохромном принтере цвета нивелируются, поэтому формы символов должны различаться.

Следует учитывать, что пакет `hyperref` автоматически загружает пакет `color` без настроек, а так как при компиляции повторная загрузка пакетов блокируется, пакет `color` с настройками нужно загружать до него.


Для работы с цветом постпроцессору, генерирующему документ, требуется драйвер с описанием палитр. Компилятор `pdflatex` автоматически загружает драйвер `pdftex` для pdf-документов или `dvips` для PostScript-документов. При работе с другими компиляторами и постпроцессорами имя драйвера [3, 4] нужно указать вместе с настройками в параметре команды `\usepackage`, загружающей пакет `color`.

По умолчанию определено 8 стандартных цветов: `black`  (черный), `red`  (красный), `yellow`  (желтый), `green`  (зеленый), `cyan`  (голубой), `blue`  (синий), `magenta`  (пурпурный) и `white`  (белый). С помощью команды


```
\definecolor{название}{палитра}{цвет}
```

автором дана возможность создать любой другой цвет в одной из палитр, т. е. моделей смешивания цветов.


Простейшей монохромной палитрой, является `gray`. Оттенки серого цвета задаются действительным числом, изменяющимся от нуля, соответствующего черному цвету, до единицы, соответствующей белому цвету.

Пример: `\definecolor{gray75}{gray}{0.75}` .

В палитре `rgb` (red-green-blue) смешиваются красный, зеленый и синий цвета. Значения компонент задает тройка чисел, изменяющихся от нуля до единицы, перечисляемых через запятую. Белому цвету соответствует набор `{1,1,1}`.

Пример: `\definecolor{MyGreen}{rgb}{0.15,0.75,0.15}` .

Базой палитры `cmuk` (cyan-magenta-yellow-black) являются голубой, пурпурный, желтый и черный цвета. Цвет определяют четыре числа, также изменяющиеся от нуля до единицы и разделенные запятыми. Белому цвету соответствует набор `{0,0,0,0}`.

Пример: `\definecolor{CT}{cmuk}{0.5,0.25,0.15,0.05}` .

Цветная верстка предусматривает изменение цветов текста и фона. Цвет текста задает декларация

```
\color[палитра]{цвет} ,
```

или команда

```
\textcolor[палитра]{цвет}{текст} .
```

Параметр дает возможность смешать цвет непосредственно для данной операции, или использовать один из уже созданных цветов, названия которых доступны при использовании специальной палитры `named`.

Приведем примеры, совместив код с изменение цвета:

```
\textcolor[rgb]{0.15, 0.6, 0.15}{\bf цвет палитры rgb}.  
{\color{CG}\bf цвет, введенный ранее}.
```

В упоминавшейся подписи к рисунку цветные символы можно сделать разными способами, так как в формулах команды `\color` и `\textcolor` тоже работают:

```
\textcolor{red}{\bullet}      ~> ●,  
$\textcolor{blue}\blacksquare$ ~> ■,  
$\color{green}\blacktriangle$ ~> ▲.
```

Цвета, показанные на рис. 7.1, определены в специальной палитре `named` с помощью команды

```
\DefineNamedColor{named}{название}{палитра}{цвет} .
```

Если пакет `color` загружен без параметра `usenames`, используя цвет, `\textcolor[named]{MidnightBlue}{\bf цвет Midnight-Blue палитры named}`, нужно указывать его название и эту палитру. Загрузка с параметром `usenames` позволяет пользоваться цветом без указания палитры: `{\bf\color{Brown} коричневый цвет}`.

Команда

```
\definecolor{название}{named}{цвет}
```

позволяет ввести синоним цвета палитры `named`. Например,

```
\definecolor{темно-бордовый}{named}{Maroon}
```

Цвет-синоним не принадлежит палитре `named` и используется как обычно: `\textcolor{темно-бордовый}{\bf темно-бордовый}`.

White		Black	
OliveGreen		Gray	
LimeGreen		PineGreen	
YellowGreen		ForestGreen	
SpringGreen		Green	
GreenYellow		SeaGreen	
Yellow		JungleGreen	
Goldenrod		Emerald	
Dandelion		BlueGreen	
Apricot		Aquamarine	
Melon		TealBlue	
Peach		Turquoise	
YellowOrange		SkyBlue	
BurntOrange		ProcessBlue	
Orange		Cyan	
RedOrange		Cerulean	
Red		Blue	
Tan		RoyalBlue	
Bittersweet		NavyBlue	
BrickRed		MidnightBlue	
Maroon		CornflowerBlue	
Mahogany		CadetBlue	
RawSienna		Periwinkle	
Brown		BlueViolet	
Sepia		RoyalPurple	
OrangeRed		Violet	
RubineRed		Plum	
WildStrawberry		Purple	
Salmon		DarkOrchid	
CarnationPink		Orchid	
Magenta		Thistle	
VioletRed		Lavender	
Rhodamine		Fuchsia	
Mulberry		RedViolet	

Рис. 7.1. Загружаемые цвета пакета color

Цвет фона можно менять в боксах и целиком на страницах. Фон боксов задают команды

```
\colorbox[палитра]{цвет фона}{текст} ,  
\fcolorbox[палитра]{цвет рамки}{цвет фона}{текст} .
```

Вторая из них создает бокс в рамке, цвет которой синтезируется в той же палитре, что и фон. Цвет задается так же, как в командах `\color` и `\textcolor`. Бокс формируется в виде строки текста, однако, используя в его аргументе окружение `minipage` или команду `\parbox`, это ограничение можно обойти.

Покажем, как в этой книге оформлялись описания команд и окружений, названия пакетов и т. д.:

```
\definecolor{фон}{rgb}{0.90,0.90,0.98}  
\colorbox{фон}{ пример серого фона }.
```

Для примера выведем цветной текст на цветном фоне:

```
\fcolorbox[YellowGreen]{Gray}{\color{yellow} \bf  
желтый текст в зеленой рамке на сером фоне }.
```

Видно, что цветные буквы имеют малую контрастность, вследствие этого текст примеров набран жирным шрифтом. По той же причине увеличена толщина рамки вокруг цветного бокса, хранящаяся в длине `\fboxrule`:

```
\setlength\fboxrule{1.5pt}.
```

По умолчанию страница не имеет фона,² но декларации

```
\pagecolor [палитра]{цвет фона} ,  
\nopagecolor
```

позволяют его установить, изменить или убрать. Фон изменяется, начиная со страницы, на которой находится начало абзаца, содержащего одну из команд, если же обе они окажутся на одной странице, фон останется неизменным. Отметим, что область действия этих деклараций не ограничивается группой.

² Говоря более строго, фон страниц прозрачен.

Глава 8

Таблицы

В первой части книги, чтобы не усложнять изложение, верстка таблиц описана без лишних деталей, тем не менее представленные в ней методы и средства позволяют создавать таблицы, имеющие сложную структуру. В этой главе рассматриваются ресурсы, предназначенные для удобства работы с таблицами, улучшения их внешнего вида и регулирования ширины, а также формирования таблиц, которые могут занимать несколько страниц.

Прежде всего скажем, что описание таблиц с большим количеством колонок можно упростить с помощью команды

```
*{число}{последовательность колонок и разделителей}
```

определенной в аргументе окружения `tabular`. Находящаяся в ее втором аргументе последовательность используется указанное число раз. Например, описание

```
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
```

можно упростить до

```
\begin{tabular}{|*{12}{c|}}
```

Чтобы между колонками не оказались две линии, первая вертикальная линия не внесена в повторяющуюся последовательность.

В особо сложных случаях можно использовать вложенные конструкции типа `*{•}{*{•}{...}}`.

8.1. Расширенный синтаксис окружения `tabular`

Стандартный пакет `array` [23] из коллекции `tools` расширяет синтаксис аргумента окружения `tabular` и определяет ряд новых команд. Он вводит дополнительные типы колонок, еще один разделитель и новую операцию *вставка*. Перечисленные дополнения сведены в табл. 8.1.

Дополнения к синтаксису окружения `tabular` Таблица 8.1

Спецификаторы колонок	Вставки
<code>b{ширина}</code>	<code>>{код}</code> — в начало ячеек
<code>m{ширина}</code>	<code><{код}</code> — в конец ячеек
<code>w{выравнивание}{ширина}</code>	Разделитель
<code>W{выравнивание}{ширина}</code>	<code>!{текст}</code>

Новый разделитель `!{●}` аналогично разделителю `@{●}` вставляет текст между колонками, но в отличие от него сохраняет между ними стандартное пустое поле. Сравните, например:

```
\begin{tabular}{|r @{} l | r !{} l |}
\hline
1 & 2 & 3 & 4 & \\
10 & 20 & 30 & 40 & \\
100 & 200 & 300 & 400 & \\
\hline
\end{tabular}
```

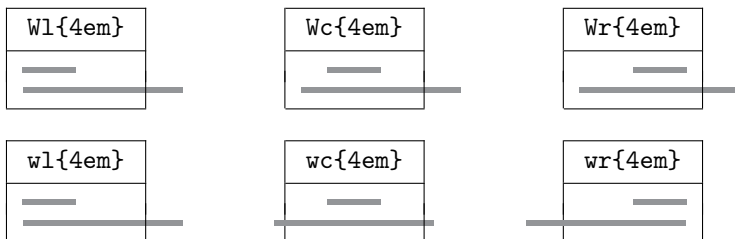
1 2	3 4
10 20	30 40
100 200	300 400

Вставки `>{код}` и `<{код}` позволяют выполнить ряд действий над всеми ячейками колонки. Первая из них должна находиться непосредственно перед спецификатором колонки, а вторая — после него. Код вставок добавляется, соответственно, в начало и конец ячеек.

Чаще всего вставки используются, чтобы изменить параметры верстки колонки. Например, с помощью конструкции `>\tt r`

текст колонки `r` можно набрать шрифтом печатной машинки, а конструкция `>$ 1 <$` заключит каждую ячейку колонки `l` в знаки доллара, поэтому она будет верстаться в математической моде. Отметим, что вставки `>{\bullet}` и `<{\bullet}` являются командами, и, воспользовавшись правилом автоматического определения аргумента, можно опустить лишние скобки вокруг знаков доллара и команды `\tt`.

Пакет `array` вводит новые типы колонок фиксированной ширины `w{\bullet}` и `W{\bullet}`, которые содержат одну строку текста. Второй их аргумент указывает ширину ячеек, а первый — тип выравнивания, задаваемый буквой: `l` — левое, `r` — правое, `c` — центрирование. Колонки различаются своим поведением при переполнении. В колонках `W` вне зависимости от типа выравнивания длинная строка выступит за правый край, а в колонках `w` она будет выравниваться, как показано ниже на примере коротких линий, уместяющихся в ячейку, и длинных, выступающих за ее границы:



Колонки `m{\bullet}` и `b{\bullet}` аналогичны стандартной колонке `p{\bullet}` и отличаются от нее только методом встраивания в строку таблицы. Их ячейки могут содержать несколько строк текста, автоматически нарезанного на строки заданной длины, выровненные справа и слева. Текст ячеек `m{\bullet}` центрируется по высоте относительно табличной строки, а у ячеек `p{\bullet}` и `b{\bullet}` в нее встраивается, соответственно, первая и последняя строка текста. Это демонстрирует пример, в котором ячейки колонок `m`, `p` и `b` содержат по три линии с длиной, равной ширине колонок, а ячейки колонок `c` — по одной:

m{4em}	c	p{4em}	c	b{4em}
=====				=====
=====	=====	=====	=====	=====
=====		=====		=====
		=====		
		=====		

Для численных данных стандартный пакет `dcolumn` [24] вводит колонку, в которой числа выравниваются по десятичному знаку:

`D{десятичный знак}{десятичный знак}{шаблон}`.

Ее спецификатор имеет три аргумента, первым из которых является символ, разделяющий целую и дробную части числа в тексте программы, а вторым — символ заменяющий его в таблице. В третьем аргументе указывается количество цифр целой и дробной частей чисел, например `5.5` или `2,5`. Колонки `D` верстаются в математической моде, поэтому их ячейки могут содержать любые математические символы.

Разберем особенности верстки таких колонок на примере:

```
\begin{tabular}{|D,.{2.3} |D,.{2,1} |D,\cdot{-1}|}
\hline
0,125 & 0,125 & 0,125 \\
10 & 10 & 10 \\
29,768 & 29,768 & 29,768 \\
\hline
\end{tabular}
```

0.125	0.125	0·125
10	10	10
29.768	29.768	29·768

В первых двух колонках сделана замена десятичной запятой на точку, а в последней — на центрированную точку. Шаблон первой колонки соответствует формату чисел, а в шаблоне второй колонки для дробной части отведено недостаточно цифр, поэтому числа вылезли за правую границу колонки.

Чтобы такое не происходило, `D` колонки имеют специальный шаблон `-1`. При его использовании компилятор определяет максимальное количество десятичных разрядов в дробных и целых

частях всех чисел и отводит его для вывода обеих частей. В нашем примере оно равно трем, поэтому применение шаблона -1 в третьей колонке привело к появлению пустого поля слева от чисел, целая часть которых состоит не более чем из двух цифр.

Таким образом, при выравнивании по десятичному знаку нужно внимательно следить за соответствием шаблона формату чисел.

Пакет `array` позволяет авторам вводить новые спецификаторы колонок, определяя для этого специальную команду:

```
\newcolumntype{символ}[число аргументов]{описание}.
```

Первый аргумент задает символ спецификатора, а параметр — число его аргументов. Описание может содержать любую комбинацию разделителей, вставок и уже известных идентификаторов. С помощью данной команды можно, например, упростить спецификатор выравнивания чисел с десятичной точкой:

```
\newcolumntype{d}[1]{D. .{#1}}.
```

Тогда при использовании нового идентификатора достаточно будет указать только шаблон вывода: `d{2.3}`.

Пакет `array` позволяет настроить правильное встраивание в текст таблиц в рамке. Как отмечалось в разд. 3.2.1, при использовании окружения `tabular` с параметром `b` или `t` в текстовую строку встраивается верх или низ рамки, а не табличная строка. Чтобы избежать этого, для верхней и нижней линии рамки введены команды

```
\firstline                      и                      \lastline .
```

Их использование дает ожидаемое поведение:

```
\begin{tabular}[*]{|c|c|}
\firstline
  a & c \\ \hline
  e & n \\
\lastline
\end{tabular}
```

a	c	текст
e	n	

8.2. Стандартные настройки

Перечислим декларации, хранящие настройки окружений, формирующих таблицы. Большая часть из них является длиной. В скобках указаны значения параметров, установленные по умолчанию.

`\arraycolsep` (5pt) — полуширина пустого поля, разделяющего колонки в окружении `array`.

`\tabcolsep` (6pt) — полуширина пустого поля, разделяющего колонки в окружении `tabular` и всех его производных.

`\doublerulesep` (2pt) — ширина зазора между соседними отчеркивающими линиями `\hline`, `\vline` и «|».

`\arrayrulewidth` (0.4pt) — толщина отчеркивающих линий `\hline`, `\cline`, `\vline` и «|».

`\arraystretch` (1) — коэффициент, на который умножается межстрочный интервал в таблице. Его значение изменяется командой `\renewcommand\arraystretch{новое значение}`.

`\extrarowheight` (0pt) — длина, добавляемая к высоте каждой строки таблицы. Вводится пакетом `array`.

8.3. Линии и рамки

8.3.1. Положение линий

В большинстве случаев интервал между табличными строками настраивать не нужно, однако иногда слишком узкий зазор между текстом и отчеркивающей линией хочется увеличить. К примеру, если ячейки содержат цифры или заглавные буквы, зазоры между ними и линиями, проведенными сверху и снизу строки, оказываются неодинаковыми:

```
\begin{tabular}{ccc} \hline
  MMM & KKK & ЖЖЖ \\ \hline
  222 & 555 & 888 \\ \hline
\end{tabular}
```

MMM	KKK	ЖЖЖ
222	555	888

Рассмотрим методы, позволяющие управлять положением отчеркивающих линий. Параметр `\arraystretch` хранит коэффициент, на который умножается межстрочный интервал. По умолчанию он равен единице. Увеличив его значение, например

```
\renewcommand\arraystretch{1.2},
```

можно раздвинуть строки. Дополнительный интервал вставляется сверху и снизу каждой строки, поэтому отчеркивающие линии симметрично разделяют их. Установка параметра в значение, меньшее единицы, уменьшит межстрочный интервал, а нулевое или отрицательное значение уберет его вовсе, однако сами строки не наложатся друг на друга.

Межстрочный интервал устанавливается для таблицы в целом, поэтому параметр `\arraystretch` не позволяет управлять отчеркиванием отдельных строк. Для этого используются другие средства.

Глубину отдельной строки можно изменить с помощью параметра команды

```
\\[длина].
```

Вставляемый дополнительный интервал сдвинет отчеркивающую линию вниз. Чтобы получить обратный эффект, т. е. сместить вниз строку относительно линии, нужно увеличить ее высоту. Для этого в одной из ячеек можно нарисовать фиктивную линию `\rule{0pt}{толщина}` нулевой ширины, и, подобрав ее толщину, добиться нужного смещения.

Пакет `booktabs` [14] предлагает более удобный метод управления положением линий. Он вводит команду

```
\addlinespace[длина],
```

которая ставится в начале строки и смещает ее вниз. По умолчанию смещение составляет `0.5em`, но параметр команды позволяет

его регулировать. Команду `\addlinespace` можно использовать не только после `\`, но и после команд, рисующих линии. Находясь между линиями, она задает расстояние между ними.

Вот как с помощью `\addlinespace` можно добиться симметричного отчеркивания в таблице предыдущего примера:

```
\begin{tabular}{ccc} \hline
\addlinespace[.5ex]
  MMM & KKK & ЖЖЖ \ \ \hline
\addlinespace[.5ex]
  222 & 555 & 888 \ \ \hline
\addlinespace[.5ex] \hline
\end{tabular}
```

MMM	KKK	ЖЖЖ
222	555	888

Чтобы показать добавленную высоту, в конце таблицы прочерчены две линии. Ширину зазора между линиями таблицы устанавливает длина `\doublerulesep`. Ее изменение повлияет на расстояние между всеми линиями. Команда `\addlinespace` позволяет регулировать зазор между отдельной парой линий.

Взамен линий `\hline` и `\cline` пакет `booktabs` предлагает целый набор линий, которые не требуют коррекции и смотрятся много лучше. Как уже говорилось в разд. 3.2.3, он вводит команды `\toprule`, `\bottomrule`, `\midrule` и `\cmidrule`. Линия, рисуемая первой командой, имеет увеличенную отбивку снизу, второй — сверху, а остальных — с обеих сторон. Вдобавок к ним определена «универсальная линия»

```
\specialrule{толщина}{верхний зазор}{нижний зазор}
```

с настраиваемой толщиной и регулируемой шириной зазоров сверху и снизу.

Покажем наглядно преимущества линий `\toprule`, `\midrule` и `\bottomrule`, вставив их в наш пример:

```
\begin{tabular}{ccc} \toprule
  MMM & KKK & ЖЖЖ \ \ \midrule
  222 & 555 & 888 \ \ \bottomrule
\end{tabular}
```

MMM	KKK	ЖЖЖ
222	555	888

В ядре \LaTeX отсутствует возможность многократного отчеркивания диапазона ячеек. Пакет `booktabs` устраняет недостачу, определяя команду

```
\morecmidrules
```

и новую длину

```
\cmidrulesep,
```

по умолчанию равную `\doublerulesep`. Чтобы дважды отчеркнуть ячейки, команду `\morecmidrules` нужно поставить между командами `\cmidrule`, например:

```
\begin{tabular}{ccc} \toprule
  MMM & KKK & ЖЖЖ \\ \cmidrule{1-1}
           \cmidrule{3-3} \morecmidrules \cmidrule{3-3}
  222 & 555 & 888 \\ \bottomrule
\end{tabular}
```

MMM	KKK	ЖЖЖ
222	555	888

С помощью разделителя `@{•}` можно регулировать ширину пустых полей вокруг вертикальных линий. Для этого в его аргумент нужно поместить команды `\vline`, окруженные пробелами, и, подобрав ширину пробелов, задать нужную отбивку. Подобно разделителю `|` команды `\vline` рисуют вертикальные линии, но как описано далее, толщину этих линий можно менять.

8.3.2. Толщина линий

Толщина линий в таблицах настраивается довольно просто.

Изменяя длину `\arrayrulewidth` можно глобально управлять толщиной линий `|`, `\hline`, `\cline`, однако задать толщину отдельной линии таким образом не удастся, так как внутри таблицы изменить `\arrayrulewidth` невозможно. С этой целью у команд `\toprule[•]`, `\bottomrule[•]`, `\midrule[•]` и `\cmidrule[•]`

предусмотрен параметр, позволяющий регулировать толщину каждой линии (см. с. 79).

Для вертикальных линий можно пользоваться командой

```
\vline width толщина ,
```

определенной в ядре ЛАТ_ЭX и имеющей нестандартный синтаксис. Ее параметр `width` устанавливает толщину рисуемой линии, а если он опущен, используется значение `\arrayrulewidth`.

Используемая в разделителях `@{•}` и `!{•}`, данная команда рисует линию по всей высоте таблицы, но ее можно использовать также в ячейках или просто в тексте, где она прочертит линию во всю высоту и глубину ячейки или строки.

8.3.3. Пересечение линий

Предваряя обсуждение верстки рамок, еще раз обратим внимание на то, что вертикальные линии загромаждают таблицы и мешают восприятию содержащихся в них данных, поэтому в своих требованиях к оформлению таблиц большинство издательств рекомендует их избегать. В частности, поэтому пакет `booktabs` не расширяет возможности вертикального отчеркивания, более того, команды `\toprule` и т. д. разбивают вертикальные линии.

Средства, имеющиеся в ядре ЛАТ_ЭX, позволяют разлиновать таблицу одиночными, двойными, тройными и т. д. линиями, однако среди них отсутствует возможность генерировать разные варианты пересечений линий. Ее предоставляет стандартный пакет `hhline` [25]. Для этого он вводит команду

```
\hhline{описание линии} ,
```

которая формирует как горизонтальную линию, так и ее сочленение с вертикальными линиями.

Линия разбивается на сегменты, описываемые в аргументе команды своеобразной мнемоникой. В простейшем случае сегменты представляют собой отрезки, отчеркивающие колонки, обозначаемые следующим образом:

- — обычная линия,
- = — двойная линия,
- ~ — отсутствие линии.

В аргументе команды `\hline` перечисляются сегменты всех колонок, например:

```
\begin{tabular}{|c|c|} \hline
  М & N & K      \\ \hline
  Т & V & W      \\ \hline
\end{tabular}
```

М	N	K
Т	V	W

В данном примере первая ячейка первой строки таблицы отчеркивается обычной линией, последняя — двойной, а средняя не отчеркивается. Зазоры между линиями возникли из-за того, что пересечение с вертикальными линиями не описано.

Как показано ниже, пересечение пары двойных линий образует квадрат, состоящий из четырех отрезков-сегментов, каждый из которых можно прочертить, или опустить.

Для управления выводом сегментов используется следующая мнемоника:

t	# — рисовать все линии,
#	t — рисовать верхнюю линию,
b	b — рисовать нижнюю линию,
	— рисовать левую или правую линию,
:	: — пропустить вертикальную линию.

Обозначение отрезков для наглядности также приведено слева.

Логика формирования линий довольно проста: перед сегментом, задающим отчеркивание ячейки, дается описание сторон «квадрата пересечений», а если оно отсутствует, рисуются только горизонтальные отрезки.

Используя перечисленные средства, можно составить много различных вариантов оформления ячейки:

```
\begin{tabular}{||c||} \hline{\bullet}
 \\ \hline{\bullet}
\end{tabular}
```

=	#=#	=	:=:	: = :	::=:::	:t=t:	t=t	b=b
=	#=#	=	:=:	: = :	::=:::	:b=b:	b=b	t=t

В примерах сверху и снизу приведены описания линий, содержащиеся в аргументах команд `\hline`. Отметим, что сегменты «:» не являются пустыми, на их месте ставится двоеточие, поэтому в пустых углах появляются точки (см. примеры в центре).

Применим теперь команду `\hline`, чтобы закончить оформление таблицы из показанного выше примера:

```
\begin{tabular}{|c|c|}
\hline|t:=:t:=:t:=:t|}
M & N & K \\
\hline||-||~|:=:|}
T & V & W \\
\hline|b:=:b:=:b:=:b|}
\end{tabular}
```

M	N	K
T	V	W

К синтаксису команды `\hline` осталось добавить, что повторяющаяся последовательность сегментов можно «свернуть» с помощью конструкции

`*{число}{последовательность сегментов}`,

повторяющей последовательность указанное число раз. Например, команду `\hline{|t:=:t:=:t:=:t|}` можно сократить до `\hline{|*3{t:=:}t|}`.

8.4. Длинные таблицы

Как говорилось ранее, таблицу, созданную окружением `tabular`, нельзя разбить на части. Чтобы снять это ограничение, стандартный пакет `longtable` [26] определяет окружение

`\begin{longtable}[выравнивание]{шаблон строки}`,

верстающее таблицу, которая может простирается на несколько страниц. Еще одним важным дополнением является возможность

сформировать подпись и присвоить таблице номер непосредственно в окружении `longtable`. Сверстанная таблица не является плавающим объектом. Ее положение в документе соответствует положению в тексте программы. Примеры длинных таблиц содержат прил. Б и В.

Параметр окружения `longtable` задает метод выравнивания таблицы в тексте и может принимать следующие значения:

- c — таблица центрируется в колонке текста (по умолчанию);
- l — таблица прижимается к левому краю;
- r — таблица прижимается к правому краю.

Окружения `tabular` и `longtable` имеют одинаковый синтаксис описания колонок и разделителей. Все методы верстки, описанные в разд. 3.2, применимы для длинных таблиц. В простейшем случае можно просто заменить окружение `tabular` на `longtable`, однако последнее имеет гораздо большие возможности.

Обычно в длинной таблице формируется преамбула, содержащая подпись и описание того, как оформляются начало и конец таблицы, а также ее разрывы на промежуточных страницах. Таблица может иметь главный заголовок, выводимый только в начале, и названия колонок, которые должны быть на каждой странице. Таким образом, начало и продолжение таблицы могут быть оформлены по-разному. Конец таблицы и ее разрывы на промежуточных страницах также могут быть разного вида. В преамбуле можно сформировать «разделы» с описаниями каждого элемента оформления.

Рассмотрим, как это делается, на следующем примере:

Пример организации длинной таблицы

Таблица 8.2

Главный заголовок		
Колонка 1	Колонка 2	Колонка 3
...
...

см. далее

Окончание табл. 8.2

Колонка 1	Колонка 2	Колонка 3
...
...

```

\begin{longtable}{|c|c|c|}
%----- преамбула -----
  \caption{Пример организации длинной таблицы}
  \label{t:LongTable} \\ \toprule
  \multicolumn{3}{c}{Общий заголовок таблицы} \\ \hline
  Колонка 1 & Колонка 2 & Колонка 3 \\ \hline
\endfirsthead
  \multicolumn{3}{r}{\small
  Окончание табл.\thetable} \\ \hline
  Колонка 1 & Колонка 2 & Колонка 3 \\
  \rule{0pt}{3ex} \\ \hline
\endhead
  \multicolumn{3}{r}{\footnotesize\it см. далее} \\ \hline
\endfoot
\endlastfoot
%----- табличные данные -----
  ..... & ..... & ..... \\
\newpage
  ..... & ..... & ..... \\
\end{longtable}

```

Чтобы разделить элементы оформления, для главного заголовка и заголовков продолжения таблицы введены команды

`\endfirsthead` и `\endhead`,

а для окончания и разрывов:

`\endlastfoot` и `\endfoot`.

Данные команды ставятся в конце описаний элементов, представляющих собой обычные табличные строки, которые компилятор подставляет в нужное место.

Разберем преамбулу нашего примера.

Таблица имеет подпись. Ее можно сформировать командами

```
\caption[краткая подпись]{подпись},  
\caption*{подпись},
```

аргументом которых служит сама подпись. В параметре первой команды можно указать сокращенный вариант подписи для списка таблиц. С командой `\caption` ассоциирован счетчик, поэтому после нее следует поставить метку `\label{•}`, ссылка на которую даст номер таблицы. При использовании команды `\caption*` формируется только подпись, сама же таблица не нумеруется и не включается в список таблиц. После подписи (или метки) нужно поставить команду перевода строки `\\`.

За подписью следует «Главный заголовок», помещенный в ячейку с объединенными колонками, а затем строка с названиями колонок. Подпись отделена от общего заголовка толстой отчеркивающей линией `\toprule`. Названия колонок отчеркнуты обычными линиями `\hline`.

Завершает главный заголовок команда `\endfirsthead`, за которой начинается описание оформления продолжения таблицы страниц. Его первая строка содержит объединенную ячейку с правым выравниванием, в которую помещен заголовок «Окончание табл.» и команда `\thetable`, выводящая текущее значение счетчика таблиц, т. е. номер таблицы. Затем следует строка с названиями колонок, отчеркнутая сверху и снизу. В последнюю ячейку добавлена команда, рисующая фиктивную линию нулевой ширины толщиной (высотой) `Зex`. Она корректирует высоту ячейки, которая слишком мала для заглавных букв и цифр, что хорошо видно на примере заголовков в начале таблицы.

Команда `\endthead` завершает заголовок промежуточных страниц. За ней следуют команды, выводящие при разрыве таблицы

отчеркивающую линию и комментарий «см. далее», помещенный в объединенную ячейку и набранный мелким курсивом. Команда `\endfoot` отделяет их от команды `\hline`, отчеркивающей последнюю строку таблицы. Заканчивает преамбулу команда `\endlastfoot`.

Порядок следования элементов оформления таблицы не важен. Хотя в нашем примере приведены все они, ни один из них не обязателен, любой можно опустить.

Верстая длинную таблицу, компилятор сам определяет место ее разбиения, если оно не указано явно с помощью команд `\newpage` или `\pagebreak`, которые можно поставить между строками, как это показано в примере. Чтобы выровнять ширину колонок на разных страницах, он разбивает таблицу на несколько частей, формирует их независимо друг от друга, а затем сравнивает и корректирует колонки. Для этого необходимо несколько проходов компиляции. По умолчанию части состоят из 20 строк, но мощность современных компьютеров позволяет увеличить их количество до ста и более, что существенно ускорит процесс формирования таблицы. Для этого нужно изменить значение счетчика

`LTchunksize`,

хранящего число строк, к примеру

```
\setcounter{LTchunksize}{200}.
```

Выше говорилось, что параметр окружения `longtable` регулирует положение таблицы относительно границ страницы или колонки текста. Более точно это можно сделать с помощью длин

`\LTleft`, `\LTRight`, `\LTpre`, `\LTpost`.

Их использование описано в [3, 26]. Еще одна длина

`LTcapwidth`

задает ширину подписи. По умолчанию ее значение равно четырём дюймам.

В длинной таблице можно делать примечания `\footnote{•}`. Если примечание относится к элементу оформления, повторя-

емому при разрыве таблицы, то его нужно разбить на метку `\footnotemark` и текст сноски `\footnotetext{•}` (см. с. 35), который должен находиться внутри окружения `longtable`.

8.5. Таблицы фиксированной ширины

Ширина обсуждавшихся ранее таблиц изменяется в зависимости от содержимого их ячеек. При необходимости ее можно зафиксировать, используя в таблице только колонки типа `p{•}` или `w{•}`.

В простейшем случае таблицу фиксированной ширины можно составить из колонок `w{•}` с разным типом выравнивания, но нужно внимательно следить за их возможным переполнением.

Колонки `b{•}`, `m{•}` и `p{•}` позволяют сверстать таблицу с ячейками, содержащими несколько строк. С помощью левой вставки и соответствующей декларации можно установить требуемое выравнивание колонок, например центрирование:

```
>{\centering} p{•}.
```

Текст ячеек будет автоматически нарезаться на строки и выравниваться, а для разрыва строк можно пользоваться командами `\\` или `\linebreak`. В этом случае, чтобы не запутаться, строки таблицы лучше заканчивать командой

```
\tabularnewline[длина],
```

специально предназначенной для этих целей. Ее параметр позволяет регулировать расстояние между строками, аналогично команде `\\[•]`. Покажем на простом примере, как сделать подобную таблицу:

```
\begin{tabular}{| >{\raggedright}m{6em}
                | wc{6em} | >{\raggedleft}m{6em} |}
\hline
    колонка 1      & 1-я строка & колонка 2
\hline
    колонка \\ 1   & 2-я строка & колонка 3 2-я строка
```

```
\tabularnewline\hline
\end{tabular}
```

колонка 1	1-я строка	колонка 3
колонка 1	2-я строка	колонка 3 2-я строка

Декларации `\raggedright` и `\raggedleft` обеспечивают левое и правое выравнивание первой и последней колонок. Средняя центрируется автоматически. В первой колонке вторая строка разорвана командой `\\`, тогда как в третьей компилятор перевел ее автоматически.

Ширина таблицы складывается из ширины колонок, равной `18em`, и окружающего их пустого поля, по умолчанию равного двум длинам `\tabcolsep` на каждую колонку. В стандартных классах эта длина равна `6pt`, поэтому ширина пустых полей составляет `36pt` или `12.636` мм.

Для создания таблиц фиксированной ширины L^AT_EX также имеет специальное окружение:

```
\begin{tabular*}{ширина}[привязка]{описание}.
```

Ширина указывается в дополнительном аргументе, а в остальном синтаксис и правила верстки `tabular*` полностью идентичны `tabular`. Окружение `tabular*` отводит под таблицу объявленную ширину и позволяет расширить таблицу, если она окажется уже, чем нужно. Если же ширина превысит объявленную, таблица вылезет за пределы отведенного ей места и наложится на окружающий текст.

Нужный размер подгоняется за счет увеличения пустого поля между колонками. Для этого предусмотрен дополнительный пробел

```
\extracolsep{длина},
```

вставляемый между колонками окружения `tabular*`. По умолчанию его длина равна нулю, но если заполнить `\extracolsep` бесконечно растяжимой длиной `\fill` расстояние между колонками автоматически подстроится под нужный размер. Чтобы изменить величину, нужно использовать разделители `@{•}` и `!{•}`.

Поясним сказанное двумя примерами:

```
\begin{tabular*}{12em}{@{\extracolsep\fill}|ccc|} \hline
  1 & 2 & 3 \\
  111 & 222 & 333 \\ \hline
\end{tabular*}
```

1	2	3
111	222	333

```
\begin{tabular*}{12em}{@{\extracolsep\fill}|c|c|c|} \hline
  1 & 2 & 3 \\
  111 & 222 & 333 \\ \hline
\end{tabular*}
```

1	2	3
111	222	333

Чтобы расстояние между колонками было одинаковым, пробел `\extracolsep` устанавливается до их описания. Колонки выглядят сбалансировано, если они не отделены друг от друга линиями. Во втором примере вертикальное отчеркивание показывает, что дополнительный пробел вставляется слева от колонки, поэтому линии смотрятся несимметрично.

Так как команда `\extracolsep` является декларацией, ширина полей между колонками, стоящими до нее не меняется. Это показывает еще один пример:

```
\begin{tabular*}{12em}{|c|c|!\extracolsep\fill}c|} \hline
  1 & 2 & 3 \\
  111 & 222 & 333 \\ \hline
\end{tabular*}
```

1	2	3
111	222	333

Стандартный пакет `tabularx` вводит окружение

```
\begin{tabularx}{ширина}[привязка]{описание} ,
```

аналогичное окружению `tabular*`. В нем определены колонки X, ширина которых подбирается, чтобы обеспечить нужный размер таблицы:

```
\begin{tabular*}{12em}{|X|X|X|} \hline
  1 & 2 & 3 \\
  111 & 222 & 333 \\ \hline
\end{tabular*}
```

1	2	3
111	222	333

Колонки X суть колонки `p{\hsize}`, ширина которых подгоняется за несколько итераций. Команда

```
\hsize ,
```

наследованная из T_EX, хранит текущую длину строки, ширину ячейки и т. д.

Так как метод создания таблиц, реализованный в окружении `tabularx`, ни в чем не превосходит и даже уступает способам, описанным в начале данного раздела, мы не будем обсуждать его более подробно и отошлем заинтересованного читателя к книгам [3, 4] и документации пакета [27].

8.6. Цвет в таблицах

Пакет `colortbl` вводит ряд команд для цветной верстки таблиц типа `tabular`. Декларация

```
\columncolor [настройка]{цвет}[левое поле]{правое поле}
```

задает фон колонки. Колонка состоит из ячеек и окружающих полей. По умолчанию декларация устанавливает фон всей колонки целиком. Два последних параметра регулируют ширину полей, заливаемых фоном.

Разберем примеры, используя введенный ранее цвет `фон`:

```
\begin{tabular}{  
  | >{\columncolor{фон}} c  
  | >{\columncolor{фон}[0pt]} c  
  | >{\columncolor{фон}[.5\tabcolsep]} [\tabcolsep]} c  
  | c  
  | >{\columncolor{фон}[1.5em]} c |}  
\hline  
  000 & 111 & 222 & 333 & 444 \\  
  555 & 666 & 777 & 888 & 999 \\  
\hline  
\end{tabular}
```

000	111	222	333	444
555	666	777	888	999

Фон колонки устанавливается с помощью вставки `>{\bullet}`, определяемой пакетом `array`. Так как пакет `colortbl` не загружает его автоматически, верстая таблицы с цветными колонками, это нужно сделать самостоятельно.

Фон первой колонки примера задан по умолчанию. У полей второй колонки фона нет, так как в параметре команды `\columncolor` указана нулевая длина. Обратите внимание: для одинаковых полей достаточно указать только одну ширину. В третьей колонке левое поле залито фоном наполовину, а правое — полностью. Чтобы сделать это использована команда `\tabcolsep`, хранящая полуширину пустого поля, разделяющего колонки.

Поля, выделяемые фоном, не связаны с полями, разделяющими колонки. Это демонстрируют две последние колонки примера. Ширина полей, залитых фоном в крайней колонке, равная `1.5em`, превышает ширину поля, разделяющего колонки. Как следствие, фон вылезает за правый край таблицы и закрывает часть колонки, расположенной слева. Следует учитывать, что раскраска не влияет на размеры, вычисляемые при формировании таблицы.

Фон табличной строки задает декларация

```
\rowcolor [наимтра] {цвет} [левое поле] {правое поле} ,
```

которая должна стоять в ее начале. По умолчанию строка выделяется целиком. Установки полей действуют на все ячейки.

Фон отдельной ячейки можно задать декларацией

```
\cellcolor [наимтра] {цвет} .
```

Ячейка раскрашивается вместе с полями, ширина которых не регулируется.

Если для фона ячейки задано несколько цветов, то цвет колонки перекрывается цветом строки, а фон команды `\cellcolor` перекрывает оба. Фон объединенной ячейки, устанавливаемый командой `\columncolor`, не изменится, если она находится в стро-

ке, фон которой тоже задан. Чтобы изменить цвет фона в этом случае, нужно применить конструкцию:

```
\multicolumn{•}{ >{\rowcolor{цвет}} •}{...},
```

в которой команда `\columncolor` замещается командой `\rowcolor`.

Начиная обсуждение раскраски отчеркивающих линий, отметим, что цвет вертикальных линий можно установить, используя возможности пакета `array`, например `!\color{red}\vline`.

Пакет `colortbl` для изменения цвета отчеркивающих линий и промежутка между ними вводит декларации:

```
\arrayrulecolor{настройка}[цвет] ,
\doublerulesepcolor{настройка}[цвет] .
```

Их нужно поставить перед таблицей. Находясь внутри нее,¹ они хорошо справляются раскраской горизонтальных линий, но с вертикальными испытывают трудности. Покажем это на примере:

```
{ \setlength\arrayrulewidth{2pt}
\arrayrulecolor{Blue} \doublerulesepcolor{фон}
\begin{tabular}{||ccc||}
  \hline\hline
  \arrayrulecolor{фон} \doublerulesepcolor{Blue}
    111 & 222 & 333      \\
  \multicolumn{3}{||c||}{444} \\
    555 & 666 & 777      \\
  \hline\hline
\end{tabular} }
\doublerulesepcolor{Black} \arrayrulecolor{Black} }
```

111	222	333
444		
555	666	777

Разберем назначение использованных команд. Сначала толщина линий `\arrayrulewidth` для наглядности установлена равной `2pt`. Затем задан синий цвет `Gray` отчеркивающих линий и светло-серый цвет `фон` для промежутков между ними. Внутри таблицы сперва нарисованы две горизонтальные линии, а потом использованы декларации, меняющие цвета местами. После

¹ Данные команды должны располагаться в начале табличных строк.

верстки таблицы черный цвет линий восстанавливается, а чтобы изменение их толщины не затронуло остальные таблицы, вся конструкция помещена в группирующие скобки.

Из примера видно, что цвета линий, отчеркивающих первую строку, соответствуют установкам, сделанным вне таблицы. После замены цветов, сделанной в начале ее первой строки, цвета горизонтальных линий, в том числе отчеркивающей первую строку снизу, меняются. Вертикальное отчеркивание второй строки определяет команда `\multicolumn`, наследовавшая сделанные изменения, поэтому его цвета также меняются. Вместе с тем, вертикальное отчеркивание третьей строки определяют параметры, действовавшие в аргументе окружения `tabular` до изменений, внесенных декларациями, находящимися внутри окружения, поэтому цвета вертикальных линий возвращаются к установкам, сделанным вне таблицы.

Богатые возможности раскраски линий дает комбинация пакетов `colortbl` и `hhline`. С помощью вставки

```
>{\arrayrulecolor{цвет}\doublerulesepcolor{цвет}},
```

которая добавляется к синтаксису команды `\hhline`, можно придать индивидуальный цвет любому сегменту линии. Не обсуждая подробно, отошлем заинтересованного читателя к документации пакетов [25, 28].

Глава 9

Настройка документа

В первой части книги описана верстка небольших документов с простой структурой типа статьи. Работа с большими текстами имеет свою специфику. Например книги, как правило, имеют указатели, библиография в них может формироваться по главам, а страницы могут иметь верхние колонтитулы. Их создание и настройка, а также другие вопросы, которых мы еще не касались, разбираются далее.

9.1. Большие документы

Большой документ удобно редактировать по частям, разбив его на несколько файлов, сводимых воедино при компиляции. Для этого нужно создать основной файл программы, содержащий преамбулу, и добавить в него команды загрузки других файлов.

Существует два варианта загрузки:

`\input{имя файла}` и `\include{имя файла}`.

При загрузке `\input` текст файла является непосредственным продолжением предшествующего текста, а команда `\include`, вставляя файл, начинает новую страницу. В файлах, загружаемых командой `\include`, можно использовать загрузки `\input`,

но вложенные загрузки `\include` запрещены. Подгружаемые файлы, в свою очередь, могут содержать команды `\input`, причем уровень вложенности таких загрузок не ограничен.

Команды `\include` должны находиться в тексте программы *после* команды `\begin{document}`. Команды `\input` могут использоваться в любом месте, но файлы, загружаемые в преамбуле, должны содержать только код. Некоторые пакеты загружают так свои дополнительные ресурсы.

Если поместить в файл команду

```
\endinput ,
```

загрузится лишь его часть, предшествующая данной команде. В основной программе ее использовать нельзя, так как компиляция завершится без создания документа.

Можно компилировать лишь часть документа. Для этого его нужно разбить на файлы, загружаемые командами `\include`, а затем в команде

```
\includeonly{список файлов}
```

указать файлы, которые следует компилировать. Данная команда должна стоять в преамбуле основного файла программы *до* команды `\begin{document}`.

В командах `\include` и `\input` имена файлов указываются по одному, а в `\includeonly` — списком через запятую. Файлы, загружаемые командами `\include`, должны иметь расширение `•.tex`, при этом их имена указываются без расширения. Команды `\input` загружают файлы с любым расширением, а если оно опущено, автоматически используется `•.tex`.

Команда `\includeonly` не влияет на загрузки `\input`. Отменить последние удастся, только отправив команды в комментарий. При этом потеряются данные о нумерации страниц и объектов, содержащихся в незагруженных файлах, в результате чего общая нумерация может сбиться. Для файлов, подгружаемых командой `\include`, генерируются собственные aux-файлы, позволяющие поддерживать правильную нумерацию, даже если сами файлы не загружаются.

Подведем итог сказанному выше. Чтобы эффективно использовать возможность компиляции документа по частям, разумно разбить его на главы, загружаемые командами `\include`, а большие главы дополнительно разбить на файлы, подгружаемые командами `\input`. Такое разбиение обеспечит правильную нумерацию в документе в целом, а использование `\includeonly` позволит избежать загрузку лишних файлов при редактировании отдельной главы.

9.2. Библиография к разделам

В некоторых книгах для каждой главы формируется собственный список литературы. В \LaTeX это делает пакет `chapterbib` в сочетании с версткой библиографии компилятором BibTeX .

Чтобы сверстать литературу к разделам, документ нужно разбить на части, загружаемые командами `\include`, каждая из которых должна содержать команды `\bibliographystyle` и `\bibliography`, указывающие стиль верстки BibTeX и библиографические базы. Если они отсутствуют, цитирование окажется невозможным, так как пакет `chapterbib` ограничивает поиск источников только загружаемым разделом.

Основная часть документа может иметь свой список литературы. Автоматически библиография генерируется только для нее, а чтобы сформировать литературу к загружаемым разделам, BibTeX придется запустить вручную для каждого из них.¹ Таким образом, для правильной обработки документа следует запустить общую компиляцию, затем сгенерировать библиографию к разделам и вновь повторить общую компиляцию.

По умолчанию для документов класса `article` литература оформляется в виде разделов `section*`, а для книг (`book`) или отчетов (`report`) — в виде нумерованных глав `chapter*`. Если библиография верстается по главам, уровень разделов со списками литературы должен быть ниже. В этом случае пакет `chapterbib` нужно

¹ BibTeX нужно запустить с `•.aux`-файлом каждого раздела.

загрузить с параметром `sectionbib`. Тогда литература основной части будет сверстана в виде главы `chapter*`, а загружаемых частей — в виде разделов `section*`.

В документации пакета `chapterbib` [29] описан ряд команд, позволяющих формировать литературу по главам без выделения их в отдельные файлы, однако использование данных ресурсов делает документ нестандартными и потому они здесь не рассматриваются.

9.3. Редактирование специальных списков

Ранее говорилось, что, собирая информацию о разделах, компилятор составляет оглавление и выводит его, если программа содержит команду `\tableofcontents`. Аналогичным образом он может сформировать списки рисунков и таблиц. Для этого используются команды

`\listoffigures` и `\listoftables`,

которые после компиляции замещаются соответствующими списками. В них включаются краткие названия рисунков и таблиц, приведенные в параметре команд `\caption`, или полные названия, если параметр отсутствует.

Списки рисунков и таблиц формируются в файлах с расширениями `•.lof` и `•.lot`. Эти файлы вместе с оглавлением, находящемся в файле с расширением `•.toc`, представляют собой специальные списки, состоящие из записей вида

`\contentsline{тип}{текст}{номер страницы}`,
`\contentsline{тип}{текст}{номер страницы}{гиперссылка}`,

в которых `текстом` является название раздела или подпись рисунка или таблицы. Третий аргумент очевиден. Четвертый аргумент, используемый для формирования гиперссылок в pdf-документах, вводит пакет `hyperref`, описанный в разд. 12.6. Типом является одно из ключевых слов `figure`, `table`, или имя команды, формирующей раздел. Приведем для ясности пример:

```
\contentsline{figure}{\numberline{2.1}\ignorespaces  
Подпись рисунка}{37}{figure.caption.11}
```

Тип определяет уровень записи в списке. В списках `lot` и `lof` все записи принадлежат верхнему уровню, но рисунки и таблицы разных глав разделяются небольшим вертикальным пробелом. Принадлежность к определенной главе позволяет уточнить команда

```
\numberline{номер} ,
```

содержащая номер объекта и выводящая его в поле определенной ширины. Команда `\ignorespaces` удаляет все следующие за ней пробелы.

Записи оглавления имеют разные уровни: к первому принадлежат записи типа `part` и `chapter`, второй и третий составляют записи `section` и `subsection`. Разделы более низкого уровня по умолчанию в оглавление не включаются, но, как указано в разд. 6.4, это можно исправить, переопределив счетчик `tocdepth`.

Информацию в специальные списки вносят команда `\caption` и команды, создающие разделы. Авторы могут добавлять в списки дополнительную информацию или корректировать параметры их верстки. Команда

```
\addtocontents{список}{текст}
```

вставляет текст в список, указанный аббревиатурой `toc`, `lof` или `lot`. Текст может содержать несколько абзацев, но обычно его составляют команды, корректирующие форматирование списка. Например, так можно отменить создание новой страницы:

```
\protect\enlargethispage*{•} \protect\nopagebreak,
```

если на нее попадает одна строка. В данном случае команда `\protect` используется, чтобы следующая за ней команда не исполнялась, а заносилась в список. Она защищает только непосредственно следующую за ней команду, поэтому ее нужно поставить перед каждой защищаемой командой. Символ `•` следует заменить минимальной длиной, достаточной, чтобы строка уместилась на текущей странице.

Еще одна команда формирует запись и добавляет ее в список:

```
\addcontentsline{список}{mun}{текст}.
```

Ее первый аргумент, как и в предыдущем случае, указывает список, в который направляется запись. Тип записи и ее текст становятся аргументами команды `\contentsline`, в которую компилятор добавит номер страницы и информацию для гиперссылки.

Для примера покажем, как вставить в список `lof` рисунок, сделанный вручную (см. конец разд. 9.4):

```
\addcontentsline{lof}{figure}{%  
  \protect\numberline{\thefigure}\ignorespaces  
  Подпись рисунка}.
```

Номер рисунка нужно поместить в команду `\numberline`, иначе оформление добавляемой строки будет отличаться от остальных пунктов списка. В данном случае следует защитить только команду `\numberline`, а команда `\thefigure` должна быть выполнена. Ее обязательно нужно поместить в скобки, чтобы составной номер рисунка попал в аргумент полностью.

В некоторых сложных случаях, после того как вся информация уже занесена в специальные списки, их форматирование можно отладить вручную. Но при компиляции, необходимой для внесения самих списков в документ, файлы обновятся, и ручная правка будет утеряна. Команда `\nofiles` помогает избежать это. Если она присутствует в преамбуле документа, запись служебных файлов, в том числе списков отменяется. Чтобы не потерять правку, когда понадобится еще что-то добавить в списки, перед тем как убрать команду `\nofiles`, рекомендуется делать их копии.

9.4. Управление плавающими объектами

Готовя к публикации статью, нет смысла тщательно подбирать место для рисунков и таблиц. Их размещением займется технический редактор издательства. При верстке собственных

документов или книг вывод рисунков и таблиц регулирует сам автор. Ранее говорилось, что параметр окружений `figure` и `tabular` позволяет указать их предпочтительное положение на странице, однако компилятор может игнорировать эти указания, если страница переполняется или оказывается недостаточно заполненной, поэтому в некоторых случаях вывод рисунков и таблиц приходится настраивать, применяя специальные методы.

Автоматическим выводом плавающих объектов управляет набор параметров, представленный в табл. 9.1. Он содержит счетчики, ограничивающие их *количество* на страницах, длины, задающие *отбивку* рисунков и таблиц между собой и в окружающем тексте, и декларации, определяющие максимальную *долю страницы*, которую они могут занять.

Настройку можно начать с этих параметров. Длины и счетчики изменяются стандартным образом, а значения долей устанавливаются с помощью команды `\renewcommand`. Например, установка счетчика `bottomnumber` в нуль:

```
\setcounter{bottomnumber}{0},
```

предотвратит появления рисунков и таблиц внизу страницы и тот же эффект даст установка в нуль доли, занимаемой плавающими объектами внизу страницы:

```
\renewcommand\bottomfraction{0}.
```

Регулировать вывод плавающих объектов помогают команды

```
\clearpage и \cleardoublepage .
```

Аналогично команде `\newpage` они заканчивают текущую страницу и при этом выводят все плавающие объекты, определенные до разрыва страницы, игнорируя некоторые ограничения на их размещение. При двусторонней печати команда `\cleardoublepage` следит за нумерацией страниц и при необходимости добавляет пустую, чтобы новая страница имела нечетный номер.

Разрывая страницу, указанные команды не заботятся о ее заполнении, поэтому стандартный пакет `afterpage` [30] вводит команду

```
\afterpage{команды} ,
```

Параметры, регулирующие вывод плавающих объектов²

Максимальное число плавающих объектов

Счетчики	Значения	Расположение
bottomnumber	1	внизу страницы
topnumber	2	вверху страницы
dbltopnumber	2	вверху колонки
totalnumber	3	на странице в целом

Максимальная доля страницы, занимаемая объектами

Команды	Значения	Расположение
<code>\bottomfraction</code>	0.3	внизу страницы
<code>\topfraction</code>	0.7	вверху страницы
<code>\dbltopfraction</code>	0.7	вверху колонки

Минимальная доля, занимаемая объектами на плавающей странице

Команды	Значения	Расположение
<code>\floatpagefraction</code>	0.5	на странице в целом
<code>\dblfloatpagefraction</code>	0.5	на странице в целом

Минимальная доля страницы, занимаемая текстом

Команда	Значение	Расположение
<code>\textfraction</code>	0.2	на странице в целом

Высота отбивки плавающего объекта

Длины	Значения	Разделение
<code>\floatsep</code>	12pt plus 2pt minus 2pt	объектов
<code>\dblfloatsep</code> ²	12pt plus 2pt minus 2pt	объектов
<code>\textfloatsep</code>	20pt plus 2pt minus 4pt	объектов и текста
<code>\dbltextfloatsep</code>	20pt plus 2pt minus 4pt	объектов и текста
<code>\intextsep</code> ²	12pt plus 2pt minus 2pt	объектов и текста

² Параметры с именами, начинающимися префиксом «dbl», применяются при наборе в две колонки.

При размере основного шрифта 12pt длины `\dblfloatsep` и `\intextsep` имеют другие величины:

$$\begin{aligned}\text{\dblfloatsep} &= 14\text{pt plus } 2\text{pt minus } 4\text{pt}, \\ \text{\intextsep} &= 14\text{pt plus } 4\text{pt minus } 4\text{pt}.\end{aligned}$$

которая заполняет страницу, а затем выполняет команды, находящиеся в ее аргументе. Если среди них имеется команда, обеспечивающая вывод плавающих объектов, они тоже выводятся. Таким образом использование связки `\afterpage\clearpage` позволяет принудительно вывести рисунки и таблицы, не нарушая заполненность страниц.

Методы, которые нужно применять для нужного размещения рисунка или таблицы, зависят от конкретной ситуации, но стоит отметить, что довольно часто вокруг плавающих объектов, созданных использованием параметра `h!`, приходится делать небольшую перестановку текста, обеспечивающую нужное пространство на странице.

Если механизм автоматического размещения плавающих объектов «победить» не удастся, рисунок или таблицу можно сделать и без их участия.

Как говорилось ранее, окружение `longtable`, создает таблицу, которая, не являясь плавающим объектом, может иметь подпись и получает номер в ряду других таблиц. Ее отбивку в тексте позволяют настроить длины `\LTpre` и `\LTpost`, первая из которых задает высоту верхнего пробела, а вторая — нижнего. По умолчанию они равны упругой длине `\bigskipamount`.

С рисунком дело обстоит сложнее, его нужно полностью форматировать вручную, как это сделано на примере рис.9.1.

*ИТЭХ
МОЖЕТ ВСЕ,
ИЛИ ПОЧТИ ВСЕ.
ИТЭХ
МОЖЕТ ВСЕ,
ИЛИ ПОЧТИ ВСЕ.*

Рис. 9.1. Рисунок, сверстаный вручную

Рисунок получился маленьким, и чтобы страница не была пустой, справа от него выведен текст, помещенный в еще одну мини-страницу.

Ниже приведен код, сверставший данный рисунок.

Поместим рисунок и подпись в мини-страницу, чтобы они встраивались в документ как целое. В нее загрузим иллюстрацию, выровняем ее и сверстаем подпись. Вдобавок к этому нужно присвоить рисунку номер и поместить его подпись в список рисунков, как это делает в окружении `figure` команда `\caption`.

```

\newcommand\figcaption[2][\lofcaption]{
  \newcommand\lofcaption{#2}
  \refstepcounter{figure}
  \addcontentsline{lof}{figure}{%
\protect\numberline{\thefigure}\ignorespaces #1}
  \small Рис. \thefigure. #2}

\begin{minipage}{.35\linewidth}
  \centering
  \includegraphics[width=.6\linewidth]{statement}\[1ex]
  \figcaption{Рисунок, сверстаный вручную}
  \label{f:HandMade}
\end{minipage}

```

Компоновка рисунков и подписей уже обсуждалась в конце разд. 3.3, поэтому рассмотрим лишь определение новой команды `\figcaption`, заменяющую стандартную команду `\caption`, которая действует только внутри плавающих объектов.

Команда `\caption` имеет аргумент, предназначенный для печати подписи в тексте, и параметр, используемый для внесения другого ее варианта в список рисунков. Если параметр опущен, оба варианта подписи совпадают. В команде `\figcaption` также предусмотрены аргумент и параметр, выполняющие те же функции. Значение параметра, используемое по умолчанию, обеспечивает команда `\lofcaption`. В нее копируется аргумент команды `\figcaption`, подставляемый вместо параметра, если он опущен.

Команда `\refstepcounter` формирует номер рисунка и ссылку на него, которую командой `\label` можно связать с меткой (см. разд. 6.4).

Команда `\addcontentsline` добавляет подпись в список рисунков (см. разд. 9.3). Слишком длинная строка с ее описанием разорвана. Чтобы в списке рисунков перед номером не появился лишний пробел, команда `\numberline` должна следовать непосредственно за фигурной скобкой. В команде `\figcaption` это обеспечивает символ комментария, стоящий непосредственно после скобки и сшивающий две строки, а также отсутствие пробелов

в начале следующей строки перед командой `\protect`, защищающей `\numberline`. Команда `\ignorespaces` убирает возможные лишние пробелы в подписи, представляющей собой текст параметра команды `\figcaption`.

Команды `\thefigure` вставляют сформированный ранее номер рисунка в аргумент `\numberline` и печатаемую подпись.

Созданная команда `\figcaption` поддерживает механизм перекрестных ссылок. Поставленная после нее команда `\label` (см. пример) связывает номер созданного рисунка с меткой, поэтому на рис. 9.1 можно ссылаться обычным образом.

9.5. Заметки на полях

Подробно структура страницы документа \LaTeX пока не рассматривалась. На рис. 9.2 показана ее разметка вместе с длинами, устанавливающими величины полей при двусторонней печати в одну колонку. В односторонней печати используется макет правых страниц. Среди приведенных длин отсутствуют лишь `\columnsep` и `\columnseprule`, хранящие ширину полей и толщину линий, разделяющих колонки при верстке в несколько колонок. Напомним, что удобный интерфейс для установки размеров всех полей дает пакет `geometry` (см. с. 27).

Помимо, собственно, текста на странице присутствуют два колонтитула и поле для заметок. Оформление колонтитулов мы рассмотрим в следующем разделе после обсуждения заметок.

Заметки на полях практически не используются в бумажной печати, так как ширина пустых полей на страницах обычно невелика. Однако в электронных документах она ограничена только шириной экрана мониторов, поэтому в заметках, наряду с подстрочными примечаниями, можно разместить пояснения.

При двусторонней печати, задаваемой параметром `twoside` во время загрузки класса документа, заметки четных страниц выносятся на левое поле, а нечетных страниц — на правое поле, как показано на рис. 9.2. Для этого ширина обоих полей должна быть одинаково большой. Увеличить их можно следующим образом.

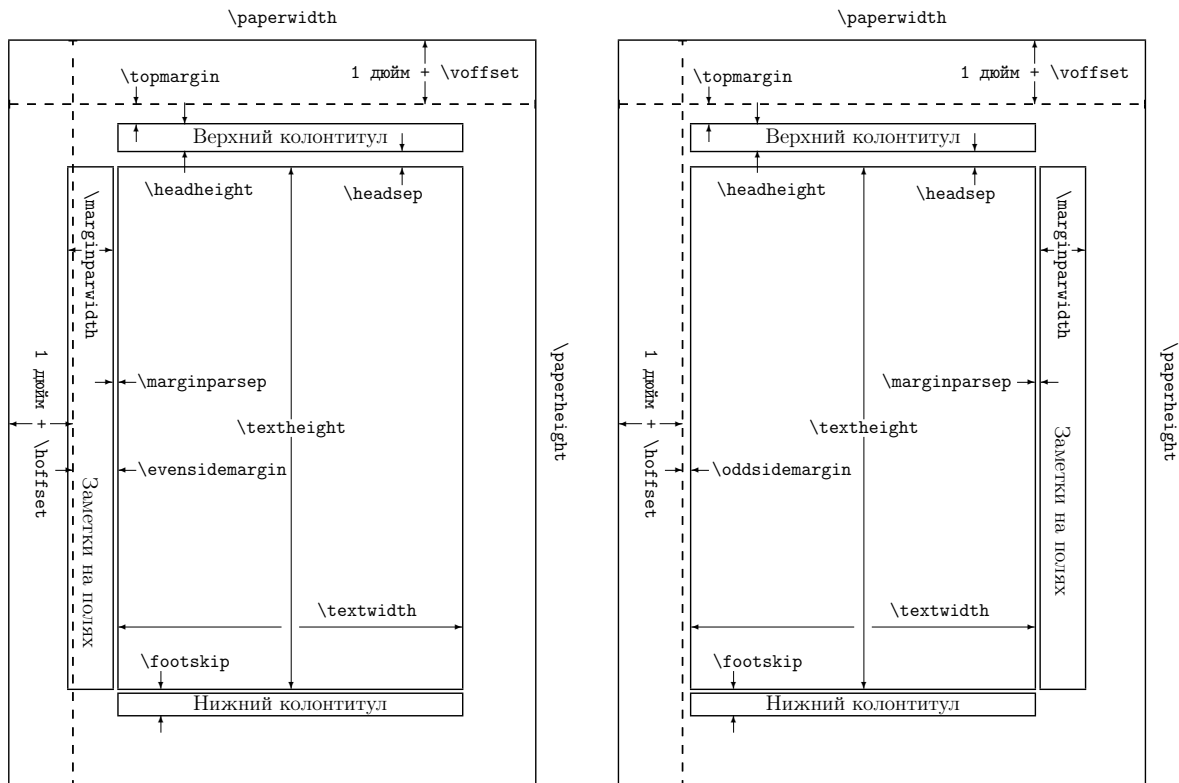


Рис. 9.2. Макет полосы набора четных и нечетных страниц

Обнулим длины `\oddsidemargin` и `\hoffset`, тогда ширина правого поля нечетных страниц будет равна ширине листа за вычетом одного дюйма и ширины текста `\textwidth`. Установим длину `\evensidemargin` равной этому значению за вычетом одного дюйма и на четных страницах получим левое поле той же ширины. Теперь можно установить ширину поля записей `\marginparwidth` не более трети или четверти ширины текста. Широкие записи смотрятся как дополнительная колонка текста, а слишком узкие плохо выравниваются.

Ширину поля заметок можно еще более увеличить, уменьшив поле переплета страниц. Для этого сдвинем влево линию отсчета полей, показанную пунктирной вертикальной линией на рис. 9.2, задав отрицательное значение длины `\hoffset`, например `\setlength\hoffset{-0.5in}`. Затем добавим к `\evensidemargin` ее удвоенное положительное значение.

Если после этих операций места для заметок не хватает, или наоборот пустые поля слишком велики, нужно скорректировать ширину листа `\paperwidth`.

При односторонней печати все заметки выводятся на правое поле страницы. При верстке в две колонки заметки к тексту левых колонок выводятся слева, а для правых — справа. Однако в документе, верстаемом в несколько колонок, вряд ли стоит делать заметки.

Декларация `\reversemarginpar` позволяет инвертировать вывод заметок. После ее выполнения заметки, выводившиеся слева, будут печататься справа и наоборот. Нормальный порядок вывода возвращает декларация `\normalmarginpar`.

Заметку на полях создает команда

```
\marginpar [текст левого поля] {текст правого поля} .
```

Ее аргумент содержит текст заметки. Текст параметра выводится в заметку, попавшую на левое поле, а если он опущен, как всегда, используется аргумент. Такое устройство команды позволяет по-разному настроить параметры верстки заметок четных и нечетных страниц.

Ширина поля, отводимого для заметок, невелика, поэтому имеет смысл уменьшить размер их шрифта. Но даже в таком виде текст, по умолчанию выравниваемый с двух сторон, может смотреться не очень хорошо из-за большого количества переносов слов. Заметка, состоящая из целых слов и выровненная по внешней стороне страницы, выглядит много лучше. Такое форматирование обеспечивают декларации `\raggedleft` и `\raggedright`. Исходя из сказанного, для верстки заметок удобно ввести новую команду:

```
\newcommand\marginote[1]{\marginpar[\footnotesize\raggedright #1]
{\footnotesize\raggedleft #1}},
```

которая на левом поле выравнивает текст слева, а на правом — справа.

Для сравнения верхняя заметка сверстана стандартной командой `\marginpar`, а следующая — новой командой `\marginote`.

В виде заметок можно оформить даже рисунок, но в документ его придется встраивать вручную, как описано в конце предыдущего раздела. Окружение `figure` в заметку вставить не удастся.

В приведенном справа примере рис. 9.4 аргумент команды `\marginpar` содержит иллюстрацию и подпись, оформленную с помощью ранее введенной команды `\figcaption`, автоматически добавившую иллюстрацию в список рисунков. Такой трюк делает документ нестандартным, но данная книга уже содержит много нестандартных решений, поэтому использование здесь «авторской» команды вполне оправданно.

Рис. 9.3. Примеры заметок на полях

Заметка, верстаемая по умолчанию, выглядит как дополнительная узкая колонка текста.

Заметки смотрятся хорошо, если не очень длинный текст набран мелким шрифтом и выровнен по внешней стороне страницы.



Рис. 9.4. Пасутся тигры на полях

Методы создания и настройки заметок обсуждаются на рис. 9.3, на котором также приведены различные примеры их оформления.

Заметки отделяются от основного текста зазором шириной `\marginparsep`.

Компилятор стремится разместить заметку на уровне текста, который она поясняет. Если команда `\marginpar` стоит внутри абзаца, первая строка заметки выводится на уровне содержащей ее строки. Если она находится между абзацами, заметка размещается на уровне последней строки предшествующего абзаца. Если же `\marginpar` окажется внизу страницы, компилятор поднимет заметку, стремясь оставить ее на текущей странице.

Такие правила действуют, когда заметок немного и они не накладываются друг на друга. В противном случае соседние заметки разделяются зазором высотой `\marginparpush`, более нижние смещаются вниз и даже могут оказаться на другой странице.

Дополнительные полезные замечания, касающиеся верстки заметок, содержит текст рис. 9.3. Чтобы сформировать поле для заметок, содержащая его страница сделана альбомной.

9.6. Настройка колонтитулов

Содержание колонтитулов регулирует декларация

```
\pagestyle{стиль страницы}.
```

Обычно ее используют в преамбуле, чтобы установить стиль для всего документа. Команда

```
\thispagestyle{стиль страницы}
```

позволяет изменить колонтитулы той страницы, на которой она находится. *Стиль страницы* имеет следующие значения:

`empty` — колонтитулы отсутствуют;

`headings` — нижний колонтитул пуст, а верхний содержит номер страницы и название раздела;

`myheadings` — нижний колонтитул пуст, а верхний содержит номер страницы и текст, задаваемый автором, который по умолчанию отсутствует;

`plain` — верхний колонтитул пуст, а нижний содержит номер страницы.

По умолчанию класс `book` использует стиль `headings`, а остальные классы — стиль `plain`.

Формирование колонтитулов — сложный процесс, состоящий из нескольких этапов, которые мы рассмотрим сначала «снизу вверх», чтобы описать участвующий в нем набор команд, а потом дадим анализ самого процесса «сверху вниз», чтобы понять, как им управлять.

Текст колонтитулов выводят команды

```
\@evenfoot{текст} ,      \@oddfoot{текст} ,  
\@evenhead{текст} ,     \@oddhead{текст} .
```

Две первые печатают нижний колонтитул, две последние — верхний.

В стиле `plain` команды `\@evenfoot` и `\@oddfoot` содержат счетчик страниц, зажатый пружинами:

```
\def\@oddfoot{\hfil\thepage\hfil},  
\let\@evenfoot\@oddfoot.
```

Как следствие, номера выводятся внизу по центру страниц. В остальных стилях команды `\@evenfoot` и `\@oddfoot` пусты, поэтому нижние колонтитулы отсутствуют.

В стилях `plain` и `empty` пусты команды `\@evenhead` и `\@oddhead`. В стилях `headings` и `myheadings` они определены как:

```
\def\@evenhead{\thepage\hfil\slshape\leftmark}.  
\def\@oddhead{{\slshape\rightmark}\hfil\thepage}.
```

Поэтому на четных страницах слева выводится номер, а справа текст, находящийся в команде `\leftmark`, печатаемый наклонным шрифтом. Такое выравнивание обеспечивает пружина, стоящая между номером и текстом. На нечетных страницах номер и текст, находящийся теперь в команде `\rightmark`, меняются местами.

Команды `\leftmark` и `\rightmark` формируются декларациями

```
\markboth{левый колонтитул}{правый колонтитул},  
\markright{колонтитул}.
```

Команда `\markright` копирует свой аргумент в `\rightmark`. Первый и второй аргументы команды `\markboth` заносятся соответственно в `\leftmark` и `\rightmark`.

Рассмотрим формирование колонтитулов в стиле `headings` при односторонней печати, которая подразумевает одинаковое оформление страниц, обеспечиваемое декларацией `\markright`.

Создавая главу в классах `book` и `report`, компилятор выполняет команду

```
\chaptermark{краткий заголовок},
```

аргументом которой является краткий заголовок главы, указанный в параметре команды `\chapter`, или ее полный заголовок, если параметр отсутствует. При односторонней печати эта команда определена как:³

```
\def\chaptermark #1 {\markright  
  {\MakeUppercase{\chaptername\ \thechapter. #1}}},
```

Декларация `\markright` заносит в команду `\rightmark` текст, печатаемый прописными буквами, состоящий из стандартного заголовка `\chaptername`, номера главы `\thechapter` и ее заголовка, находящегося в аргументе `#1` и отделенного от номера точкой.

В классе `article` ту же функцию выполняет команда

```
\sectionmark{краткий заголовок},
```

```
\def\sectionmark #1 {\markright  
  {\MakeUppercase{\thesection\quad #1}}},
```

участвующая в создании раздела. Она помещает в `\rightmark` номер и заголовок раздела, печатаемый прописными буквами.

Таким образом, текст колонтитулов обновляется для каждой новой главы, или раздела.

³ Определения воспроизводятся не буквально, часть системных команд заменена знакомыми читателю аналогами, описанными ранее.

Отметим, что ранее не встречавшиеся команды

```
\MakeUppercase{текст} и \MakeLowercase{текст}
```

переводят строчные буквы в прописные и наоборот.

При двусторонней печати четные и нечетные страницы оформляются по-разному, поэтому определение команды `\chaptermark` меняется следующим образом:

```
\def\chaptermark #1 {\markboth  
  {\MakeUppercase{\chaptername\ thechapter. #1}}{}}
```

Теперь она задает текст левого колонтитула и «очищает» правый, текст которого формирует команда `\sectionmark` во время создания разделов. Определение последней

```
\def\sectionmark #1 {\markrigh  
  {\MakeUppercase{\thesection. #1}}},
```

претерпевает лишь слабое изменение, заключающееся в замене пробела `\quad` точкой. В результате выполнения обеих команд левый колонтитул содержит названия и номера глав, а правый — названия и номера разделов.

В классе `article` меняется определение команды `\sectionmark`:

```
\def\sectionmark #1 {\markboth  
  {\MakeUppercase{\thesection\quad #1}}{}}
```

и вводится команда

```
\subsectionmark{краткий заголовок},
```

```
\def\subsectionmark #1 {\markrigh  
  {\MakeUppercase{\thesubsection\quad #1}}},
```

выполняемая при создании раздела `\subsection`. Теперь левый колонтитул содержит названия и номера разделов, а правый — названия и номера подразделов.

Разберем теперь, какую последовательность действий влечет изменение стиля страниц. Прежде всего отметим, что стиль, заданный в преамбуле, можно корректировать «по месту» в тексте, но как правило, во всем документе выдерживается единый стиль. Приведенный ниже пример является исключением, сделанным для иллюстрации настройки колонтитулов.

9.6.1. Стиль `empty`

Декларация `\pagestyle{empty}` очищает аргументы команд `\@evenfoot`, `\@oddfoot`, `\@evenhead` и `\@odhead`, и более они автоматически не обновляются. Назовем такое действие *аннулированием*. Если после этого отредактировать команды вручную, внесенный в них текст появится в колонтитулах. Например, после установки

```
\makeatletter
```

```
\def\@evenhead{\itshape\hfil-- Четные страницы --\hfil}  
\makeatother
```

вверху по центру четных страниц будет выводиться надпись

– *Четные страницы* –.

Чтобы не загромождать приведенные далее примеры, мы опустим в них команды, меняющие статус символа \mathcal{O} .

9.6.2. Стиль `plain`

Декларация `\pagestyle{plain}` аннулирует аргументы команд `\@evenhead` и `\@odhead`, а команды `\@evenfoot` и `\@oddfoot` настраивает на вывод номера по центру внизу страницы, как это описано ранее.

Формат номеров страниц в колонтитулах задает декларация

```
\pagenumbering{формат}.
```

Форматами служат имена команд, перечисленных в табл. 6.2. Стандартом являются арабские цифры, но иногда для вводной части книг используются большие и малые римские цифры.

Декларация `\pagenumbering` начинает новый отсчет страниц, устанавливая значение их счетчика в единицу и меняя формат его вывода.

Если в преамбуле документа задать `\pagenumbering{roman}`, а перед первой главой — `\pagenumbering{arabic}`, то страницы вводной части будут нумероваться малыми римскими цифрами, а нумерация первой главы начнется с обычной единицы.

Редактируя `\@evenfoot` и `\@oddfoot`, можно настроить остальные аспекты вывода номеров, например напечатать их мелким шрифтом в левом углу на четных страницах и в правом на нечетных, как это делают определения:

```
\def\@evenfoot{\small \thepage \hfil},
\def\@oddfoot{\small \hfil \thepage}.
```

9.6.3. Стиль `headings`

Декларация `\pagestyle{headings}` аннулирует аргументы команд `\@evenfoot` и `\@oddfoot`, а в аргументы команд `\@evenhead` и `\@oddhead` помещает команды `\leftmark` и `\rightmark` вместе с номерами страниц в формате, прежде заданном `\pagenumbering`.

Описанный выше механизм формирования текста команд `\leftmark` и `\rightmark` обеспечивает следующее оформление книги. Первая страница главы содержит только номер, а на следующих четных страницах в колонтитулах печатается также название и номер главы. До создания первого раздела колонтитулы нечетных страниц содержат только их номера, а потом к ним добавляется номер и название раздела.

Разберем настройку оформления колонтитулов на следующем примере. Пусть в левом углу четных страниц выводится номер страницы, по центру маленьким шрифтом прописными буквами — название главы, а в правом углу — ее номер римскими цифрами. Пусть на нечетных страницах номер выводится справа, по центру печатается название раздела, а слева его номер. Сами же колонтитулы отделяются от текста сплошной линией, как показано на двух следующих страницах.

Опишем настройку таких колонтитулов. Так как отчеркивание затрагивает печать колонтитулов в целом, прежде всего переопределим обеспечивающие ее команды:

```
\def\@evenhead{\underline{
\makebox[\textwidth]{
\thepage\hfill\leftmark}}}
```

```
\def\@oddhead{\underline{
      \makebox[\textwidth]{
        \rightmark\hfill\thepage}}}
```

Для отчеркивания применена команда `\underline{•}`, имеющая особенность, существенную в данном случае: в подчеркнутом тексте не действуют пружины, необходимые для выравнивания колонтитула. Поэтому аргументом `\underline` служит команда `\makebox`, создающая бокс с шириной, равной ширине страницы. В бокс вставлены номер страницы и текст колонтитула, разделенные жесткой пружиной.

Теперь отредактируем команды, формирующие текст колонтитулов,

```
\def\chaptermark #1{
      \markboth{\small\MakeUppercase{#1}
        \hfill ГЛ.\, \Roman{chapter}}{}}
\def\sectionmark #1{
      \markright{\$, \thesection \hfill #1}}
```

В команду `\leftmark` декларация `\markboth` помещает команды `\small` и `\MakeUppercase`, обеспечивающие нужное оформление названия главы `#1`, жесткую пружину `\hfill`, отделяющую его от заголовка ГЛ. и номера главы, печатаемого командой `\Roman`. Заголовок и номер разделены тонким пробелом `\,`.

Текст правого колонтитула, который декларация `\markright` помещает в команду `\rightmark`, начинается символом `§`, тонким пробелом и номером раздела `\thesection`, за которыми следуют жесткая пружина и название раздела `#1`.

В результате «сборки» колонтитулов в командах `\@evenhead` и `\@oddhead` названия глав и разделов центрируются пружинами, отжимающими номера страниц, глав и разделов на края страниц. Жесткие пружины применяются, чтобы нивелировать действие мягких пружин `\hfil`, которые неявно используются, если допустить автоматическое выравнивание текста в боксе `\makebox`.

9.6.4. Стиль `myheadings`

Декларация `\pagestyle{myheadings}` аннулирует аргументы команд `\@evenfoot` и `\@oddfoot`, а в аргументы команд `\@evenhead` и `\@oddhead` помещает команды `\leftmark` и `\rightmark` вместе с номерами страниц в формате, прежде заданном `\pagenumbering`. При этом текст из команд `\leftmark` и `\rightmark` убирается, а его автоматическое обновление при создании новых разделов отключается. В итоге автоматически формируемые колонтитулы содержат лишь номера страниц, разнесенные по левым и правым углам на четных и нечетных страницах, но авторам дана возможность вручную задать текст команд `\leftmark` и `\rightmark` с помощью деклараций `\markboth` и `\markright`.

Стиль `myheadings` позволяет настроить колонтитулы разделов, формируемых командами `\chapter*`, `\section*` и `\subsection*`. При их создании текст команд `\leftmark` и `\rightmark` аннулируется, так как по умолчанию их название в колонтитулы не вносится, и при необходимости это автор должен сделать сам.

Приведем пример. Согласно правилам российской полиграфии введение и заключение книги не нумеруются, поэтому их удобно оформлять, используя раздел `\section*`. Тогда, установив стиль `myheadings`, после команды `\section*` нужно поставить команду `\markboth{•}{•}` с названием раздела `•`, которое попадет в колонтитулы⁴.

Не забудьте вновь включить стиль `headings` непосредственно *перед* созданием следующего нумерованного раздела.

Настройка колонтитулов — специфическая задача, возникающая довольно редко. Описанные средства позволяют с ней справиться, но если их окажется недостаточно, воспользуйтесь пакетом `fancyhdr` [31], позволяющим верстать колонтитулы любой сложности.

⁴ В книге [2] ее рекомендуется ставить без пробела после первого слова раздела.

9.7. Настройка списков

В первой части книги подробно разобрана верстка библиографии, описано создание нумерованных и ненумерованных списков. За кадром остались вопросы, связанные с настройкой формата вывода списков, разбираемые далее.

Описанные в разд. 5.1 списки `itemize`, `enumerate` и `description` имеют общие настройки, так как они суть модификации одного и того же окружения `list`, служащего основой еще целого ряда окружений. Поэтому сначала мы разберем устройство `list`, а потом посмотрим, как оно используется для создания списков и других конструкций.

Окружение `list` представляет собой список с довольно сложной внутренней структурой, представленной на рис. 9.5, настраиваемой большим числом параметров:

```
\begin{list}{метка}{настройки}
    список
\end{list}
```

По умолчанию элементы списка используют метку, заданную в первом аргументе окружения. Во втором аргументе можно настроить команду, выводющую метку, и установить значения параметров, управляющих версткой списка.

Параметры представляют собой длины, задающие горизонтальные и вертикальные отбивки элементов списка. Все они показаны на рис. 9.5, а их значения приведены в табл. 9.2–9.3.

Предваряя анализ настроек, отметим, что в вертикальной отбивке участвуют упругие длины, поэтому высота отбивок может меняться в довольно широких пределах.

В разд. 5.1 говорилось, что списки `itemize` и `enumerate` имеют четыре уровня вложенности, а у `description` их шесть. Величины длин варьируются в зависимости от уровня вложенности, размера шрифта, которым набирается список, и размера основного шрифта документа. Различные комбинации этих параметров представлены в табл. 9.2.

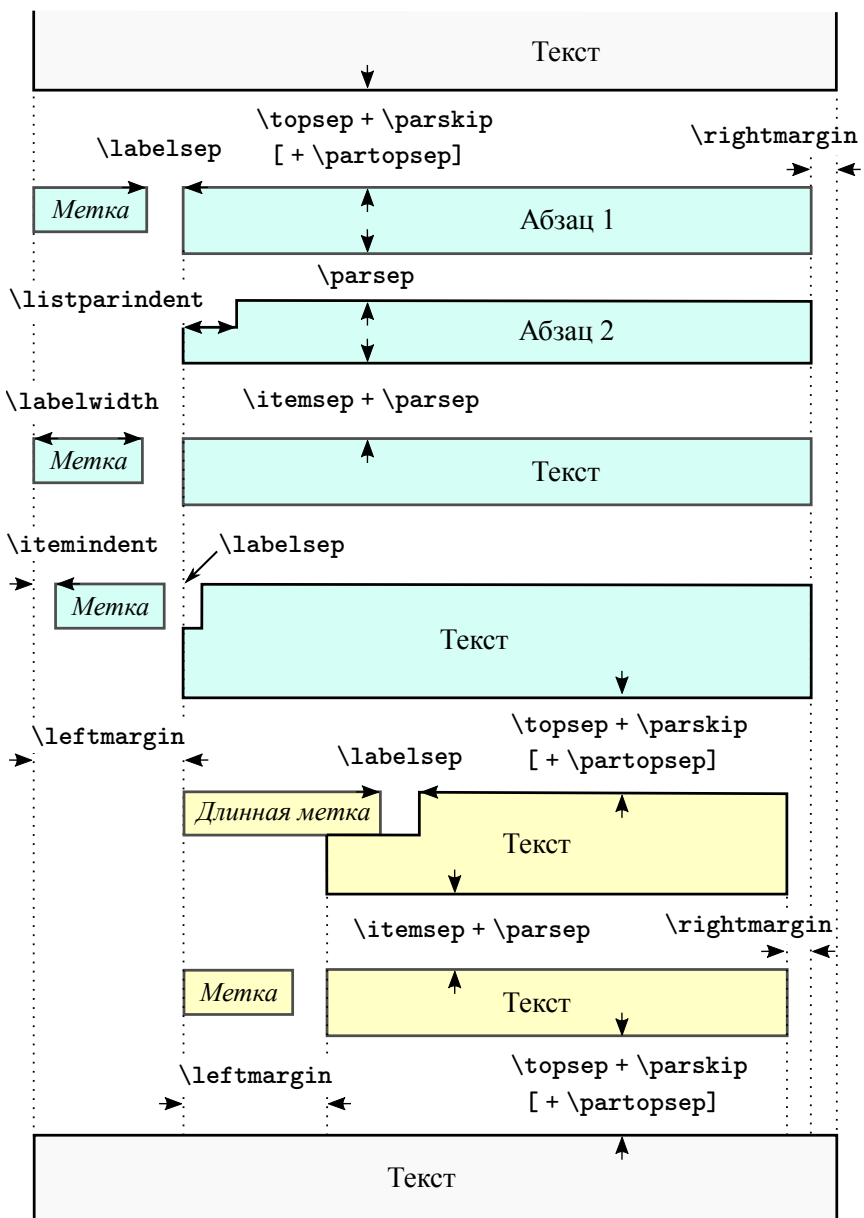


Рис. 9.5. Формат стандартных списков

Таблица 9.2

Параметры настройки стандартных списков: длины вертикальных отбивок

Кегль	Длина	<code>\@listi</code>	<code>\@listii</code>	<code>\@listiii</code>	<code>\small</code>	<code>\footnotesize</code>
12pt	<code>\topsep</code>	10pt +4pt -6pt	5pt +2.5pt -1pt	2.5pt ±1pt	9pt +3pt -5pt	6pt ±2pt
	<code>\parsep</code>	5pt +2.5pt -1pt	2.5pt ±1pt	0pt ±0pt	4.5pt +2pt -1pt	3pt +2pt -1pt
	<code>\partopsep</code>	3pt ±2pt	3pt ±2pt	1pt +0pt -1pt	<code>\partopsep</code>	<code>\partopsep</code>
	<code>\itemsep</code>	5pt +2.5pt -1pt	<code>\parsep</code>	<code>\topsep</code>	<code>\parsep</code>	<code>\parsep</code>
11pt	<code>\topsep</code>	9pt +3pt -5pt	4.5pt +2pt -1pt	2pt ±1pt	6pt ±2pt	4pt ±2pt
	<code>\parsep</code>	4.5pt +2pt -1pt	2pt ±1pt	0pt ±0pt	3pt +2pt -1pt	2pt ±1pt
	<code>\partopsep</code>	3pt ±1pt	3pt ±1pt	1pt ±0pt	<code>\partopsep</code>	<code>\partopsep</code>
	<code>\itemsep</code>	4.5pt +2pt -1pt	<code>\parsep</code>	<code>\topsep</code>	<code>\parsep</code>	<code>\parsep</code>
10pt	<code>\topsep</code>	8pt +2pt -4pt	4pt +2pt -1pt	2pt ±1pt	4pt ±2pt	3pt ±1pt
	<code>\parsep</code>	4pt +2pt -1pt	2pt ±1pt	0pt ±0pt	2pt ±1pt	2pt ±1pt
	<code>\partopsep</code>	2pt ±1pt	2pt ±1pt	1pt +0pt -1pt	<code>\partopsep</code>	<code>\partopsep</code>
	<code>\itemsep</code>	4pt +2pt -1pt	<code>\parsep</code>	<code>\topsep</code>	<code>\parsep</code>	<code>\parsep</code>

Таблица 9.3

Параметры настройки стандартных списков: длины горизонтальных отбивок

Общие				Одна колонка		Две колонки	
Команда	Длина	Команда	Длина	Команда	Длина	Команда	Длина
<code>\labelsep</code>	0.5em	<code>\leftmarginii</code>	2.2em	<code>\leftmargini</code>	2.5em	<code>\leftmargini</code>	2em
<code>\bibindent</code>	0.5em	<code>\leftmarginiii</code>	1.87em	<code>\leftmarginv</code>	1em	<code>\leftmarginv</code>	0.5em
<code>\parskip</code> ⁵	0pt +1pt	<code>\leftmarginiv</code>	1.7em	<code>\leftmarginvi</code>	1em	<code>\leftmarginvi</code>	0.5em

⁵Вертикальный размер.

Декларации `\@listi`, `\@listii` и `\@listiii` задают длины для списков трех уровней вложенности, а начиная с четвертого, используются настройки третьего уровня. Приведем для примера определение одной из них:

```
\def\@listii {\leftmargin \leftmarginii
               \labelwidth \leftmarginii
               \advance\labelwidth -\labelsep
               \topsep 4.5pt plus 2pt minus 1pt
               \parsep 2pt plus 1pt minus 1pt
               \itemsep \parsep}.
```

В нем используется синтаксис присваивания и изменения значений Т_ЕX:

```
команда = значение,
\advance команда by значение.
```

Команда может быть длиной или счетчиком, значение можно задать явно или другой командой. Очень часто знак равенства и ключевое слово `by` опускаются. Например, в приведенном определении длине `\leftmargin` присваивается значение длины `\leftmarginii`, то же значение присваивается длине `\labelwidth`, которая потом уменьшается на длину `\labelsep`.

В табл. 9.2 значения длин приведены для шрифта размером 10pt, 11pt и 12pt вместе с величинами, задаваемыми при его уменьшении декларациями `\small` и `\footnotesize`. Команда, поставленная вместо значения, указывает на эквивалентность длин. Например, длине `\itemsep` присваивается значение `\parsep` или `\topsep`, а длина `\partopsep` при уменьшении шрифта не меняется.

Назначение длин показывает рис. 9.5.

Элемент списка может состоять из нескольких абзацев, разделенных зазором высотой `\parsep`. Чтобы элементы хорошо выделялись зрительно, их отбивка увеличивается на высоту `\itemsep`. Еще более широкий зазор отделяет список от окружающего текста. Его высота складывается из длин `\topsep` и `\parskip`. Последняя добавляется к межстрочному интервалу между абзацами

текста как внутри, так и вне списков. Ее величина представлена в нижней части таблицы вместе с горизонтальными длинами. Если список стоит перед абзацем или следует за ним, т. е. перед ним или после него находится пустая строка или заменяющая ее команда `\par`, отбивка увеличивается на высоту `\partopsep`.

Верстка строк управляется неупругими длинами, величины которых выражаются в единицах `em`, привязанных к текущему размеру шрифта. Первая строка элемента списка, содержащая метку, верстается по собственным правилам, а остальные — как обычный текст. Длина строк ограничена слева и справа пустыми полями, шириной, соответственно `\leftmargin` и `\rightmargin`. Как показано на рис. 9.5, поля списков первого уровня отсчитываются от краев страницы, или границ предшествующего абзаца, поля вложенных списков — от границ предыдущего списка.

По умолчанию `\rightmargin` устанавливается в нуль, а случаи, в которых это значение изменяется, мы оговорим особо. Во вложенных списках длина `\leftmargin` получает значение `\leftmargin \bullet` , где \bullet обозначает римскую цифру от i до vi , соответствующую уровню вложенности ($\bullet \equiv i-vi$). Величины длин `\leftmargin \bullet` приведены в табл. 9.3.

Метка выносится на левое пустое поле и помещается в бокс, шириной `\labelwidth`, равной длине `\leftmargin` за вычетом длины `\labelsep`, отделяющей метку от текста элемента. Если метка оказывается длиннее отведенного размера, ширина бокса увеличивается так, чтобы она полностью в него уместилась. В этом случае метка вклинивается в текст элемента, отделяясь от него зазором `\labelsep`, как показано внизу рис. 9.5.

Строка с меткой имеет отступ шириной `\itemindent`, величина которого добавляется к длинам `\labelwidth` и `\labelsep`. Если он не равен нулю, текст первой строки отступает от границы пустого поля на ту же длину. Элемент списка может насчитывать несколько абзацев текста. Их отступ регулирует длина `\listparindent`. Значения `\itemindent` и `\listparindent` по умолчанию устанавливаются в нуль.

На входе в окружение `list` вычисляется уровень вложенности списка, и если он не превышает шести, выполняется соответствующая декларация

```
\@list•{установка длин} , • ≡ i–vi,
```

устанавливающая величины перечисленных ранее длин. Команды `\@list•` и значения длин задает класс документа. Уровень вложенности хранит счетчик `\@listdepth`.

Метка формирует и выводит команда

```
\makelabel{метка} ,
```

определенная в окружении `list`. Ее аргументом являются текст или символ метки и команды форматирования, настраиваемые отдельно для каждого типа списка.

В качестве метки может выступать значение счетчика, например номер в списке `enumerate`. В этом случае декларация

```
\usecounter{счетчик}
```

привязывает счетчик к списку. Она должна присутствовать во втором аргументе окружения `list`, а счетчик должен быть определен до входа в окружение. Декларация `\usecounter` устанавливает счетчик в нуль, а каждая команда `\item` обновляет его значение с помощью команды `\refstepcounter`, обеспечивающей возможность его использования в системе перекрестных ссылок (см. разд. 1.4.6).

Разберем теперь определения окружений ЛАТ_EX, созданных на основе списка `list`, но прежде скажем, как менять их настройки *после* определения. Если между командами `\begin{имя окружения}` и `\item`, т. е. перед первым элементом списка, отредактировать значения указанных выше длин, формат вывода списка изменится, при этом сделанные изменения не затронут остальные списки. Чтобы настроить верстку всех списков, нужно в преамбуле отредактировать декларации `\@list•` ($• \equiv i-vi$).

Представленные далее определения содержат компактные нотации Т_EX, используемые в оригинальных кодах. Структура

окружений воспроизводится корректно, но не буквально. Часть команд, несущественных для понимания конструкции в целом, не приводится. Любознательный читатель найдет их в документации ядра ЛАТ_EX [20] или коде стандартных классов.

9.7.1. Окружение `itemize`

Окружение `itemize` помечает элементы символами, которые печатают команды

```
\newcommand\labelitem•{\labelitemfont символ}, •≡i-iv
```

Их задает класс документа, подобно тому, как это было сделано на с. 153 в определении команды `\itemlabelset`.

В стандартных классах метками служат символы

- `\textbullet` (i), — `\textendash` (ii),
- * `\textasteriskcentered` (iii), · `\textperiodcentered` (iv),

а шрифт `\labelitemfont` соответствует шрифту `\normalfont`.

Окружение `itemize` определено следующим образом:

```
\newenvironment{ itemize }
{ Контроль уровня вложенности.
  \begin{list}{\labelitem•}
    {\def\makelabel##1{\hss ##1}
  }}{\end{list}}
```

Разберем его устройство.

Сначала вычисляется уровень вложенности окружения, и если он не превышает четырех, формируется окружение `list` с меткой `\labelitem•`. Команда `\makelabel` во втором аргументе окружения `list` определена так, чтобы метки, прижимаясь к краю бокса, выравнивались по правому краю. Обозначение `##1` передает компилятору обращение к аргументу команды, а не к аргументу окружения. Команда `\hss`, наследованная из Т_EX, является бесконечно растяжимым и сжимаемым клеем, имеющим свойства пружины. На выходе из окружения `itemize` выполняется команда `\end{list}`.

Таким образом, окружение `itemize`, по сути, является окружением `list` с описанными выше настройками.

Отметим, что обычные пружины `\hfil` и `\hfill` не сминают клей `\hss`, поэтому они не могут регулировать выравнивание меток. Чтобы получить такую возможность, нужно переписать команду `\makelabel`. Например, определение

```
\renewcommand\makelabel[1]{\hfil\bfseries #1},
```

сохранив правое выравнивание, сделает метку жирной и позволит команде `\item`[символ `\hfil`] сместить ее в центр бокса или к его левой границе при использовании пружины `\hfill`.

9.7.2. Окружение `enumerate`

В нумерованных списках класс документа настраивает метки в два этапа. Сначала определяются команды вывода счетчиков:

```
\renewcommand\theenumi{\arabic{enumi}},  
\renewcommand\theenumii{\alph{enumii}},  
\renewcommand\theenumiii{\roman{enumii}},  
\renewcommand\theenumiv{\Alph{enumiv}},
```

а затем они используются для задания меток:

```
\newcommand\labelenumi{\theenumi.},  
\newcommand\labelenumii{(\theenumii)},  
\newcommand\labelenumiii{\theenumii.},  
\newcommand\labelenumiv{\theenumiv.}.
```

Кроме этого определяются префиксы, выводимые перед номерами элементов вложенного списка, в ссылках `\ref`:

```
\renewcommand\p@enumii{\theenumi},  
\renewcommand\p@enumiii{\theenumi(\theenumii)},  
\renewcommand\p@enumiv{\p@enumiii\theenumiii}
```

В соответствии с ними и определениями команд `\theenum` ссылка на элемент списка четвертого уровня будет состоять из номера элемента первого уровня, указанного арабскими цифрами, стоя-

щей за ним в круглых скобках строчной буквы, соответствующей номеру элемента второго уровня, за ними следует номер элемента третьего уровня, набранный малыми римскими цифрами, и, наконец, прописная буква, соответствующая номеру самого элемента, например 2(a)іС.

Список `enumerate` определен аналогично списку `itemize`:

```
\newenvironment{ enumerate }
{ Контроль уровня вложенности.
  \begin{list}{\labelenum●}
    {\usecounter{enum●} \def\makelabel##1{\hss ##1}
  }}{\end{list}},
```

с той лишь разницей, что метки задают команды `\labelenum●` ($\bullet \equiv i-iv$), а второй аргумент окружения `list` содержит дополнительно команду `\usecounter`, связывающую список со счетчиком `enum●`. Таким образом, все сказанное ранее о настройках списка `itemize` напрямую относится и к `enumerate`.

9.7.3. Окружение `description`

Окружения `itemize` и `enumerate` вводятся в ядре \LaTeX , список `description` определяет класс документа. Для этого создается команда вывода метки

```
\newcommand* \descriptionlabel[1] { \hspace\labelsep
                                   \normalfont\bfseries #1}
```

и использующее ее окружение

```
\newenvironment{ description }{
  \begin{list}{ }{ \labelwidth Opt
    \itemindent -\leftmargin
    \let\makelabel\descriptionlabel
  }}{\end{list}}.
```

Разберем устройство окружения `description`.

Оно не имеет собственного контроля вложенности, а окружение `list` ограничивает ее шестым уровнем. Окружение `description` предназначено для создания списка терминов, служащих мет-

ками. У каждого элемента термин свой, поэтому по умолчанию метки нет, первый аргумент окружения `list` пуст, а ширина бокса для метки устанавливается в нуль.

Отрицательная длина отступа `\itemindent` смещает печать метки влево на ширину пустого поля `\leftmargin`. Команда `\makelabel` приравнивается команде `\descriptionlabel`, которая для вывода метки устанавливает жирный шрифт и вставляет перед ней пробел шириной `\labelsep`.

Окружение `list` помещает метку в бокс, ширина которого, сначала равная нулю, после формирования метки оказывается равной сумме длины `\labelsep` и ширины термина. Пробелы, находящиеся перед термином, игнорируются, поэтому при печати он смещается в начало бокса, ширина которого фиксирована, поэтому за термином формируется пробел длиной `\labelsep`.

В результате действия всех команд термин, смещенный на границу пустого поля, печатается жирным шрифтом, а пробел `\labelsep` отделяет его от последующего текста.

9.7.4. Окружение `thebibliography`

Список литературы также формирует окружение `list`, использующее вместо команды `\item` команду `\bibitem`. Окружение `thebibliography` задает класс документа, определяющий его следующим образом:⁶

```
\newenvironment{ thebibliography }[1]{
  \section*{\refname}
\markboth{\MakeUppercase\refname}{\MakeUppercase\refname}
  \begin{list}{ \@biblabel{\arabic{enumiv}} }{
    \settowidth\labelwidth{\@biblabel{#1}}
    \leftmargin \labelwidth
    \advance\leftmargin \labelsep
```

⁶ Здесь приведено определение класса `article.cls`. В классах `book.cls` и `report.cls` команда `\refname` заменяется на `\bibname`.

```

\usecounter{enumiv}
\renewcommand\p@enumiv{ }
\renewcommand\theenumiv{\arabic{enumiv}}
\@openbib@code
}}{\end{list}}

```

Разберем его устройство.

Сначала команда `\section*` выводит заголовок библиографии в документ, команда `\markboth` заносит его в колонтитулы, переводя все буквы в заглавные. Затем формируется окружение `list`. В качестве метки ему передается команда `\@biblabel`, печатающая номер источника арабскими цифрами. Счетчик `enumiv` используется для нумерации списка литературы.

Во втором аргументе окружения `list` задаются параметры верстки источников. Ширина бокса с меткой `\labelwidth` устанавливается равной ширине текста, находящегося в аргументе `#1` окружения `thebibliography`, печатаемого командой `\@biblabel` (см. разд. 5.2.1 и с. 151). Ширина левого поля `\leftmargin` устанавливается равной сумме длин `\labelwidth` и `\labelsep`. Счетчик `enumiv` привязывается к списку литературы, а формат его вывода в ссылках `\ref` корректируется. Для этого его префикс `\p@enumiv` делается пустым, а команда `\theenumiv` настраивается на печать номеров источников арабскими цифрами.

В конце выполняется команда `\@openbib@code`. По умолчанию она является пустой, но загрузка класса документа с параметром `openbib` изменяет ее, чтобы дополнить настройки:

```

\renewcommand\@openbib@code{ \parsep 0pt
\advance\leftmargin \bibindent
\advance\itemindent -\bibindent
\advance\listparindent \itemindent }

```

В результате ее действия ширина левого поля увеличивается на длину `\bibindent` (см. табл. 9.3) и в соответствие с новым значением приводятся отступы `\itemindent` и `\listparindent`. Кроме этого параметр `openbib` изменяет команду `\newblock`, используемую стилями BibTeX для выделения логических блоков

в описаниях источников (см. пример на с. 241). По умолчанию она вставляет небольшой пробел перед новым блоком

```
\newcommand\newblock{
    \hspace{.11em plus .33em minus .07em}},
```

а после переопределения

```
\renewcommand\newblock{\par}
```

начинает блок с новой строки.

9.7.5. Окружения `quote`, `quotation`, `verse`

Еще три окружения также построены на основе списка `list`. Их также водит класс документа. Среди них самое простое определение у окружения `quote`:

```
\newenvironment{ quote }{
    \begin{list}{}{\rightmargin \leftmargin}
    \item\relax }{\end{list}}
```

Метка по умолчанию в нем не устанавливается, а ширины левого и правого полей делаются равными, что дает одинаковую отбивку текста слева и справа. На входе в окружение `list` выполняется команда `\item\relax`, поэтому текст окружения `quote` представляет собой абзацы одного элемента списка. Но если в нем присутствуют другие команды `\item[•]`, формируются новые элементы. Уровень вложенности «списков» `quote` не должен превышать шести.

В окружении `quotation` добавляется отступ абзацев, и уменьшается величина зазора между ними:

```
\newenvironment{ quotation }{
    \begin{list}{}{ \listparindent 1.5em
        \itemindent \listparindent
        \rightmargin \leftmargin
        \parsep 0pt plus 1pt }
    \item\relax }{\end{list}}
```

Для верстки стихов в ЛАТ_EX предусмотрено ранее не обсуждавшееся окружение `verse`. Оно не имеет аргументов. Верстаемое стихотворение разбивается на строфы пустыми строками, а в строфе строки разделяются командами `\\`. Приведем для примера пару двустиший Ренаты Мухи:

Один Осьминог подошел к Осьминогу
И в знак уваженья пожал ему ногу.
Мама — Зебра, папа — Лось.
Как им это удалось?

```
\begin{verse}
  Один Осьминог подошел к Осьминогу\\
  И в знак уваженья пожал ему ногу.

  Мама --- Зебра, папа --- Лось.\\
  Как им это удалось?
\end{verse}
```

Окружение `verse` устроено наиболее сложно:

```
\newenvironment{verse}{
  \let \\ \@centercr
  \begin{list}{}{ \itemsep 0pt
    \itemindent -1.5em
    \listparindent \itemindent
    \rightmargin \leftmargin
    \advance\leftmargin 1.5em }
  \item\relax }{\end{list}}
```

Прежде всего команда перевода строки заменяется командой `\@centercr`, выполняющей эту функцию по-иному. Она заканчивает строку командой `\par`, убирая при этом все пробелы, предшествующие тексту следующей строки.

В окружении `list` длина `\itemsep` устанавливается в нуль. Длины отступов `\itemindent` и `\listparindent` корректируются так, что первые строки абзацев смещаются влево на `1.5em`. Ширина левого поля делается на `1.5em` больше правого. Учитывая, что

каждая строка набираемой строфы фактически становится абзацем, левое и правое поля оказываются одинаковыми, и только продолжение длинных строк смещается вправо на 1.5em.

В остальном свойства окружений `quote`, `quotation` и `verse` одинаковы, поэтому все сказанное о `quote` относится также и к двум другим окружениям.

9.7.6. Окружение `trivlist` и его клоны

Самостоятельной модификацией списка `list` является окружение `trivlist`:

```
\begin{trivlist} текст \end{trivlist}.
```

Оно не имеет аргументов и использует настройки списков, описанные в начале раздела. Контроль уровня вложенности в нем отсутствует, команды `\list` не выполняются, поэтому настройки всех окружений одинаковы. Длины `\labelwidth`, `\leftmargin` и `\itemindent` устанавливаются в нуль, а длина `\parsep` приравнивается к `\parskip`. Строки внутри и вне окружения имеют одинаковую длину, абзацы — одинаковую вертикальную отбивку.

Окружения `center`, `flushleft` и `flushright`, выравнивающие текст, строятся на основе `trivlist`:

```
\newenvironment{center}{ \begin{trivlist}
  \centering \item\relax }\end{trivlist}}
\newenvironment{flushleft}{ \begin{trivlist}
  \raggedright \item\relax }\end{trivlist}}
\newenvironment{flushright}{ \begin{trivlist}
  \raggedleft \item\relax }\end{trivlist}}
```

Оно также является основой окружений `verbatim` и `verbatim*`:

```
\newenvironment{verbatim}{ \begin{trivlist}
  \item\relax Прембула }\end{trivlist}},
\newenvironment{verbatim*}{ \begin{trivlist}
  \item\relax Прембула }\end{trivlist}},
```

преамбула которых содержит дополнительный набор команд для буквального воспроизведения текста.

Среди математических конструкций окружение `trivlist` используют теоремы `newtheorem` и `newtheorem*`. Ядро `ЛATEX` и пакет `amsthm.sty` определяют их по-разному. Их код имеет сложную структуру и поэтому здесь не анализируется.

9.8. Л^AT_EX по-русски

Основные настройки, необходимые для редактирования текстов, написанных по-русски, делает пакет `babel`, загружаемый с параметром `russian`, который задает правила переноса слов, русифицирует стандартные названия, добавляет принятые у нас нотации тригонометрических и гиперболических функций и другие ресурсы верстки. Вместе с тем он не может полностью адаптировать `ЛATEX` к традициям российской полиграфии. Основные ее требования стоит обсудить особо.

Часть из них касается рисунков и таблиц. Автоматическое размещение в документе с помощью механизма плавающих объектов, применяемое `ЛATEX`, плохо согласуется с необходимостью их расположения непосредственно перед или после первой ссылки на них, принятой в российских изданиях. Это требование можно обеспечить, используя при создании плавающих объектов параметр `h!`, а если это не помогает, применить принудительный вывод связкой команд `\afterpage\clearpage`.

В русских текстах номер и подпись к рисунку или таблице должны разделяться точкой, тогда как в стандартных классах для этой цели служит двоеточие. Здесь на помощь приходит пакет `caption` [32], позволяющий настроить все аспекты вывода подписей. В частности, чтобы уменьшить шрифт подписи и установить в качестве разделителя точку загрузите его с параметрами:

```
\usepackage [font=small, labelsep=period] {caption}.
```

Компактные таблицы являются наиболее удобным и эффективным методом организации справочного материала. В российской полиграфии принято отчеркивать колонки вертикальными линиями, однако в ГОСТах такого требования нет. Чаще всего вертикальное отчеркивание загромождает таблицу и увеличивает ее ширину, поэтому большинство зарубежных издательств его не использует. В рукописях статей рекомендуется отчеркивать лишь верх, низ и заголовок таблиц. Современные полиграфические технологии позволяют использовать наглядное фоновое выделение колонок или строк, примененное в данной книге, однако в журнальных публикациях это пока не принято.

Еще одним важным отличием является прямое начертание греческих букв в формулах, принятое в российской полиграфии, и курсивное — в L^AT_EX. В данной книге использованы стандартные (для L^AT_EX) шрифты **ComputerModern**, в которых как латинские, так и греческие буквы печатаются курсивом. В свободном доступе имеются шрифты **AMS Euler** Американского математического общества, содержащие прямой математический курсив с начертанием и насыщенностью близкой к шрифтам **ComputerModern**. Греческие буквы этих шрифтов подключает пакет `upgreek`⁷ из коллекции `was`.

Буквы прямого начертания введены командами с именами, начинающимися с префикса `up`, например

$$\backslash upxi \rightsquigarrow \xi \text{ или } \backslash upvarpi \rightsquigarrow \varpi.$$

Чтобы формулы, содержащие прямые буквы, можно было бы копировать в другие документы, имеет смысл дать этим командам стандартные имена. Для этого удобно применить упоминавшуюся ранее конструкцию T_EX [1] `\let\VA`, присваивающую команде `\V` значение команды `\A`. Загрузив пакет `upgreek` и сделав в преамбуле документа присваивания

⁷ Загрузка пакета без параметра подключает шрифты **AMS Euler**, а при загрузке с параметром `Symbol` используются греческие буквы шрифта **Adobe Symbol** с начертанием и насыщенностью, близкими к шрифтам **Times**.

```
\let\alpha\upalpha  
\let\beta\upbeta...
```

получим прямое начертание строчных букв, выводимых стандартными командами. Заглавные греческие буквы имеют прямое начертание по умолчанию, поэтому их можно не переопределять.

Последнее замечание касается обсуждавшегося ранее десятичного знака. В российской полиграфии им традиционно служит запятая, но никакими ГОСТами это не утверждено. ГОСТ 8.417—2002 («Единицы величин») пользуется только запятой, но сам по себе в качестве десятичного разделителя ее не устанавливает. ГОСТ 6.20.1—90 («Электронный обмен данными в управлении, торговле и на транспорте») и ГОСТ 2.004—88 («Общие требования к выполнению конструкторских и технологических документов на печатающих и графических устройствах вывода ЭВМ») допускают оба варианта.

Десятичную часть числа можно отделять либо точкой, либо запятой, но нельзя использовать с этой целью оба знака в одном документе! В \LaTeX от десятичной запятой следует отказаться, так как рукописи статей должны содержать десятичную точку, а копирование текста из одного документа в другой может «смешать» десятичные знаки.

Глава 10

Стили цитирования

В первой части обсуждалась верстка списка литературы, цитирование источников, а также создание и редактирование библиографических баз. В данной главе мы опишем настройку оформления ссылок с помощью пакетов `cite` и `natbib` и разберем особенности стиля цитирования автор-год.

10.1. Оформление ссылок

Пакет `cite` [18] предназначен для работы с нумерованными списками литературы, взаимодействуя с которыми, команда `\cite` по умолчанию заключает ссылки в квадратные скобки и выводит их в строку как обычный текст. Пакет `cite` настраивает стиль оформления ссылок, генерируемых этой командой, и позволяет вывести их в виде верхних индексов. Основная настройка осуществляется при его загрузке с помощью следующих параметров:

`sort, nosort` — (не) сортировать номера источников в ссылках;

`compress, nosocompress` — (не) заменять последовательные номера диапазоном;

`space`, `nospace` — заменить тонкий пробел, разделяющий номера, в ссылках на обычный, или убрать его;

`adjust`, `noadjust` — (не) регулировать пробел перед ссылками;

`nobreak` — запретить разрыв строки внутри ссылок;

`superscript`, `super` — оформить ссылки в виде верхних индексов;

`move`, `nomove` — (не) переносить вперед знаки препинания, поставленные после ссылок, оформленных в виде верхних индексов;

`ref` — добавлять слово «Ref.» перед номером в ссылках, имеющих уточняющую информацию, т. е. оформленных с помощью команды `\cite[●]{●}`;

`biblabel` — привести формат вывода номера источника в списке литературы, задаваемый командой `\@biblabel`, в соответствии с форматом вывода ссылок командой `\cite`.

При загрузке пакета без параметров списки номеров в ссылках сортируются, сокращаются до диапазонов и выводятся в квадратных скобках в строку, при этом команда `\cite` оптимизирует пробелы, стоящие перед ссылками.

При использовании параметра `superscript` или `super` ссылки оформляются в виде верхних индексов без скобок, пробелы перед ними игнорируются, а следующие за ними знаки препинания переносятся вперед, например

`Arseneau \cite{Arseneau-cite} . \rightsquigarrow Arseneau.`¹⁸

Для вывода каждого элемента ссылки пакет `cite` вводит собственную команду. Переопределение этих команд позволяет изменить оформление ссылок.

Команды `\citeleft` и `\citeright` выводят скобки, охватывающие ссылку, поэтому комбинация

`\renewcommand\citeleft(\renewcommand\citeright)`

заменит квадратные скобки круглыми.

Команды

`\citeform{•номер•} cite`

выводят номера ссылок. На месте символов • можно поставить любые команды, меняющие формат вывода. Например, переопределение

`\renewcommand\citeform[1]{/#1/}`

оформит ссылки в виде [/1/, /3/] или ^{1/}/_{3/}.

Команды `\citepunct`, `\citedash` и `\citemid` разделяют элементы в ссылках, содержащих список номеров, или уточняющую информацию. По умолчанию диапазон номеров разделяется коротким тире `\endash`, их список — запятой и небольшим пробелом, а уточняющая информация отделяется от номера запятой и обычным пробелом. Пробелы обеспечивают возможность автоматического перевода строки внутри длинной ссылки. Переопределение

`\renewcommand\citepunct{,}`

сохранит запятую между номерами источников, но ссылка станет «монолитной» и будет встраиваться в строку как целое.

Команда `\CiteMoveChars` хранит список знаков препинания, которые переносятся вперед, если стоят после ссылки, оформленной в виде верхнего индекса. По умолчанию в него входят точка, запятая, двоеточие и точка с запятой.

Команда `\OverciteFont` устанавливает шрифт ссылок, оформляемых в виде верхних индексов.

Команды

`\citen{список ярлыков}` и `\citenum{список ярлыков}`

выводят номера ссылок в строке без скобок:

Авторы работы [`\citen{article}`]... \rightsquigarrow Авторы работы [12]...,
Обычно они используются, если основные ссылки оформляются в виде верхних индексов. В таких случаях скобки приходится ставить вручную.

Команды полностью эквивалентны, а `\citenum` добавлена для совместимости с пакетом `natbib`, рассматриваемом в следующем разделе.

10.2. Стил ь цитирования автор-год

Для верстки нумерованной библиографии вполне достаточно обсуждавшихся выше ресурсов стандартных стилей BibTeX и пакета cite. При работе со списками литературы, упорядоченными по фамилиям авторов, наиболее эффективен пакет `natbib`, поддерживающий в полном объеме также и нумерованные списки. Ввиду своей универсальности он рекомендован к использованию многими издательствами, поэтому опишем его основные возможности, ограничившись лишь теми, что связаны с цитированием источников.

Как правило, пакет `natbib` используется вместе со специальными стилями BibTeX, позволяющими реализовать все его возможности. Вместе с ним устанавливаются стили `plainnat`, `abbrnat`, и `unsrnat`, выполняющие те же функции, что и стандартные стили, описанные в разд. 5.2.2. Для оформления библиографии по ГОСТу пакет `gost`, рассмотренный там же, добавляет к ним аналоги `gost2008n`, `ugost2008n`, `gost2008ns`, и `ugost2008ns`.

Отличительной особенностью стилей, предназначенных для работы с пакетом `natbib`, является формирование в окружении `thebibliography` преамбулы, в которой определяются команды, используемые для форматирования источников. Набор команд зависит от стиля оформления списка литературы.

Покажем, как будут выглядеть список литературы и ссылка на источник, представленные на с. 130, при использовании пакета `natbib` и стиля `ugost2008n`:

В документе:

... в книге Кнут (1993) описаны ...

Список литературы

Кнут Дональд Е. Все про TeX. – Протвино: АО RDT_EX, 1993. – С. 575. – ISBN: 5-900614-01-8.

.... в книге \cite{Knuth-TeXbook-1993} описаны ...

```
\begin{thebibliography}{99}
\def\selectlanguageifdefined#1{
  \expandafter\ifx\csname date#1\endcsname\relax
  \else\selectlanguage{#1}\fi}
\providecommand*\BibEmph}[1]{#1}
\providecommand*\BibDash{
  \ifdim\lastskip>0pt\unskip\nobreak\hskip.2em plus 0.1em\fi
  \cyrdash\hskip.2em plus 0.1em\ignorespaces}

\bibitem[Кнут (1993)]{Knuth93}
  \selectlanguageifdefined{russian}
  \BibEmph{Кнут~Дональд~Е.} Все про {\upshape\TeX}. \BibDash
  \newblock Протвино~: А0 RD\TeX, 1993. \BibDash
  \newblock С.~575. \BibDash
  \newblock ISBN:~5-900614-01-8.
```

В примере показана только часть команд, содержащихся в преамбуле. Видно, что по сравнению с кодом, генерируемым обычными стилями BibTeX, преамбула и описание источника выглядят гораздо сложнее.

Ссылка на источник, выводимая в документ, в примере она состоит из фамилии автора и года выхода книги, заносится в параметр команды \bibitem. Стиль оформления ссылок задается при загрузке пакета многочисленными параметрами, из которых наиболее часто используются:

authoryear, numbers — стиль цитирования автор-год (используется по умолчанию), или номер ссылки;

round, square, curle, angle — оформление ссылок в круглых, квадратных, фигурных или угловых скобках;

comma, semicolon, colon — разделение списка ссылок запятой, точкой с запятой (используется по умолчанию) или двоеточием;

`super` — оформление нумерованных ссылок в виде верхних индексов;

`sort&compress`, `sort`, `compress` — сортировка ссылок в нумерованных списках и замена последовательности номеров диапазонами (второй и третий параметры выполняют только одну из указанных операций);

`sectionbib` — генерация библиографии по разделам с уровнем не ниже, чем глава (для этого также нужно загрузить пакет `chapterbib`).

Таблица 10.1
Стили оформления ссылок пакета `natbib`

Загрузка с параметром <code>authoryear</code> (библиография автор-год)		
Стили	<code>\citep{•}</code>	<code>\citet{•}</code>
<code>plain</code>	<code>[#, #,...]</code>	автор [год1, год2],...
<code>cospar</code>	<code>/#, #,.../</code>	автор /год1 год2/,...
<code>nature</code>	<code>текст^{#,#,...}</code>	автор ^{год1,год2} ,...

Загрузка с параметром `numbers` (нумерованная библиография)

Стили	<code>\citep{•}</code>	<code>\citet{•}</code>
<code>plain</code>	<code>[#, #,...]</code>	автор [#, #],...
<code>cospar</code>	<code>/#, #,.../</code>	автор /# #/,...
<code>nature</code>	<code>текст^{#,#,...}</code>	автор ^{#,#} ,...

Для любой библиографии

Стили	<code>\citep{•}</code>	<code>\citet{•}</code>
<code>plainnat</code>	<code>[автор, год1, год2,...]</code>	автор [год1, год2],...
<code>agu</code>	<code>[автор, год1, год2;...]</code>	автор [год1, год2];...
<code>egu, agms</code>	<code>(автор, год1, год2;...)</code>	автор (год1, год2);...
<code>dcu</code>	<code>(автор; год1, год2;...)</code>	автор (год1, год2);...
<code>kluwer</code>	<code>(автор год1, год2,...)</code>	автор (год1, год2),...

Пакет `natbib` предоставляет несколько предустановленных стилей оформления ссылок, перечисленных в табл. 10.1, которые можно задать или изменить в любом месте программы с помощью декларации

`\citestyle{стиль цитирования}`.

Среди них присутствует и стандартный вывод нумерованного списка в квадратных скобках.

Ссылку, содержащую фамилию автора, можно заключить в скобки полностью, как показано во второй колонке таблицы, или оставить фамилию в тексте, указав в скобках год или номер публикации (см. третью колонку). С помощью двух типов команд пакет `natbib` обеспечивает оба варианта оформления ссылок.

Ссылки, в которых год или номер заключаются в скобки, а фамилии из них выносятся, генерируют команды

```
\citet[перед][после]{список ярлыков} ,  
\citet*[перед][после]{список ярлыков} .
```

Первая включает в ссылку фамилию первого автора, а вторая — весь их список. С помощью параметров `перед` и `после` части ссылки, заключенной в скобки, можно вставить дополнительную информацию. Если задан один параметр, она добавляется в конец ссылки. Приведем примеры ссылок в стиле `egu`.¹

```
\citet{jon90}           ~> Jones et al. (1990)  
\citet*{jon90}         ~> Jones, Baker, and Williams (1990)  
\citet{jon90, jam91}   ~> Jones et al. (1990); James (1991)  
\citet{jam90-1, jam90-2, jam91} ~> James (1990a, 1990b, 1991)  
\citet[see][chap. 2]{jam91} ~> James et al. (see 1991, chap. 2)  
\citet[review][ ]{jon90} ~> Jones et al. (review 1990)
```

Общие ссылки коллектива авторов собираются вместе. Если библиография содержит несколько работ коллектива, выпедших в свет в одном году, они сортируются по названию и к году добавляются буквы `a`, `b`, `c` и т. д.

Дополнительно определены команды `\Citet` и `\Citet*`, которые все фамилии авторов, включая титульные приставки, начинают заглавными буквами, например:

```
\citet{dRob98} ~> della Robbia (1998)  
\Citet{dRob98} ~> Della Robbia (1998)  
\Citet*{dRob98} ~> Della Robbia and Williams (1998)
```

¹ В основу положены оригинальные примеры пакета `natbib`.

Ссылки, заключенные в скобки, формируют команды

```
\citep[перед] [после]{список ярлыков},
```

```
\citep*[перед] [после]{список ярлыков}.
```

Как и в предыдущем случае, первая генерирует ссылку с фамилией первого автора, а вторая — с полным их списком. Первый параметр используется для вставки дополнительной информации *перед* ссылкой, а второй — *после* нее. Они тоже имеют аналоги `\Citep` и `\Citep*`. Для сравнения покажем те же ссылки в скобках:

```
\citep{jon90}           ~> (Jones et al., 1990)
\citep*{jon90}         ~> (Jones, Baker, and Williams, 1990)
\citep{jon90, jam91}   ~> (Jones et al., 1990; James, 1991)
\citet{jam90-1, jam90-2, jam91} ~> (James, 1990a, 1990b, 1991)
\citep[see] [chap.2]{jam91} ~> (see James et al., 1991, chap. 2)
\citep[review] []{jon90} ~> (review Jones et al., 1990)
\Citep{dRob98}         ~> (Della Robbia, 1998)
```

Все перечисленные выше команды имеют варианты с суффиксом `al`, верстающие ссылки без скобок, например:

```
\citealt*{jon90}       ~> Jones, Baker, and Williams, 1990
\citealt{jam90-2, jam91} ~> James, 1990b, 1991
\citealp[see] [(chap.2)]{jon91} ~> see Jones et al., 1991, (chap. 2)
\Citealp{dRob98}      ~> Della Robbia, 1998
```

Оформление ссылок по умолчанию, т. е. без явного указания командой `\citetstyle`, зависит от того, как верстается библиография. Стиль `plain` используется в ссылках на нумерованную библиографию. Если список литературы верстается в стиле авторгод с помощью стилей пакета `natbib`, ссылки генерируются в стиле `plainnat`. Оформление ссылок на библиографию, сверстанную стилями пакета `gost`, соответствует стилю `egu`.

Для унификации цитирования следует пользоваться обычной командой `\cite`. При загрузке пакета `natbib` без параметра `numbers` она становится полным аналогом команды `\citet` в стиле `plainnat`, а при загрузке с этим параметром — команды `\citep` в

стиле `plain`. Отметим, что у нее появляется второй параметр, поэтому она становится не совсем стандартной, но если пользоваться только одним, копирование текста манускрипта в документы, компилируемые без пакета `natbib`, не вызовет проблем.

При загрузке пакета с параметром `super` ссылки, генерируемые командой `\cite`, нумеруются и оформляются в виде верхних индексов без скобок. Команда

```
\citenum{список ярлыков}
```

позволяет при необходимости вывести их в строке:

Основополагающая работа Ландау~[\citenum{Landau-JETP-1953}] и ряд последующих статей\cite{...}	Основополагающая ра- бота Ландау [2] и ряд по- следующих статей ³⁻⁷ ...
--	--

Мы перечислили лишь часть ресурсов пакета `natbib`, дающего возможность настроить все аспекты верстки библиографии и ссылок. Остальные подробно описаны в его документации [33].

Стоит отметить, что в настоящее время активно разрабатывается пакет `biblatex`, в будущем претендующий стать основным средством работы с библиографией. Его использование предполагает работу с компиляторами `xelatex`, или `lualatex` и программой `biber`, которые пока не являются стандартом. Заинтересованных читателей мы отсылаем к обширной документации пакета [34], насчитывающей более трехсот страниц.

Глава 11

Предметный указатель

Важной составной частью книг является предметный указатель, содержащий отсылки к отсортированным по алфавиту понятиям, терминам и т. п. Элементы указателя значительно упрощают поиск нужной информации. Приведем его пример:

<code>\begin{theindex}</code>	Предметный указатель
.....
<code>\item лазеры, 12, 39, 115</code>	Л
<code>\subitem газové, 87</code>	лазеры, 12, 39, 115
<code>\subsubitem CO₂, 96</code>	газовые, 87
<code>\subsubitem He-Ne, 89</code>	CO ₂ , 96
<code>\subitem твердотельные, 15</code>	He-Ne, 89
.....	твердотельные, 15
<code>\end{theindex}</code>

Слева показано окружение `theindex` с элементами указателя, а справа — его вид в документе.

В тексте программы описания элементов указателя помещаются в команды `\index{•}`. Компилятор \LaTeX собирает их при обработке документа. Сортировкой данных и составлением указателя занимается специальная программа, в качестве которой в

настоящее время чаще всего выступает `xindy`. Алгоритмы сортировки обладают большой гибкостью, позволяющей не только учесть специфику любого алфавита, но наладить сортировку сразу для нескольких языков. Программа `texindy` запускает `xindy` с настройками, позволяющими разбирать специфический синтаксис описаний элементов указателя, и передает ей данные, собранные компилятором `LATEX`.

Блок-схема процесса генерации предметного указателя показана на рис. 11.1. Стрелками показаны направления потоков информации при взаимодействии `LATEX` и `xindy`.

На первом этапе `LATEX` собирает записи, находящиеся в тексте программы, дополняет их номерами страниц и помещает в файл с расширением `.idx`. Затем запускается программа `texindy`, передающая его вместе с настройками программе `xindy`, которая сортирует записи, формирует окружение `theindex` и записывает его в файл с расширением `.ind`. В ходе последующих компиляций `LATEX` вставляет указатель в печатный документ.

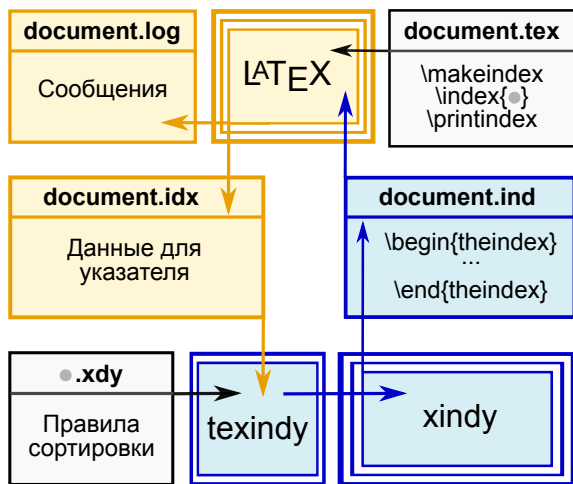


Рис. 11.1. Создание предметного указателя с помощью `xindy`

Настройка `xindy` для составления англо-русского указателя описана в прил. Г. Здесь же мы рассмотрим средства \LaTeX , обеспечивающие его создание.

Прежде всего, необходимо загрузить пакет `makeidx`, определяющий команду

```
\index{описание записи},
```

предназначенную для ввода описаний элементов указателя в программу. По умолчанию \LaTeX игнорирует эти команды, поэтому сбор описаний необходимо инициировать командой

```
\makeindex,
```

которая должна стоять в преамбуле. Чтобы обеспечить корректность номеров страниц в указателе, команды `\index{•}` следует помещать непосредственно рядом с терминами. Их можно использовать внутри окружений с рисунками и таблицами.

При компиляции \LaTeX добавляет номера страниц к описаниям элементов, преобразует команды `\index{•}` в команды

```
\indexentry{описание элемента}{номер страницы}
```

и направляет их в файл с расширением `•.idx`.

Чтобы указатель появился в печатном документе, программа должна содержать команду

```
\printindex,
```

которая замещается окружением `theindex`. При ее отсутствии указатель в документ не вставляется.

Окружение `theindex` представляет собой перечень с вложенностью до трех уровней. Записи первого уровня помечаются декларацией `\item`, второго — `\subitem`, а третьего — `\subsubitem`. Сортированные записи разбиваются по алфавиту на рубрики.

Описания элементов в аргументах команд `\index{•}` определяются синтаксисом программы `MakeIndex`, используемой для составления указателей англоязычной литературы. Они представляют собой текст, разбитый на ряд полей, в простейшем случае состоящий лишь из самого термина, например `\index{лазеры}`.

Поля разных уровней разделяются восклицательным знаком:

```
... \index{лазеры!газовые}...  
... \index{лазеры!газовые!He-Ne}...  
... \index{лазеры!газовые!CO$_2$}...
```

Каждое поле, в свою очередь, может разбиваться на дополнительные поля, позволяющие настроить сортировку термина и, при необходимости, выделить основную ссылку на него.

Для сортировки можно определить заголовок, который ставится перед термином и отделяется от него символом @. Например, элемент `\index{von Beethoven}` попадет в категорию «V», а с заголовком (`\index{Beethoven@von Beethoven}`) он будет отсортирован по фамилии. Заголовок также используется для описания символов: `\index{греческие буквы!alpha@α}`. Описание, содержащее математическую формулу без заголовка, может вызвать ошибку сортировки, автоматически прекращающую составление указателя.

Стандартные команды L^AT_EX позволяют изменить шрифт, которым печатается термин. Они не влияют на сортировку. Например, записи второго уровня в представленном выше примере указателя можно набрать курсивом:

```
\index{лазеры!\textit{газовые}! He-Ne}.
```

Шрифтовое оформление термина не затронет оформления номера страницы. Для этого предназначено еще одно дополнительное поле, следующее за термином и отделяемое от него символом |. Оно позволяет управлять выводом ссылки. В простейшем случае это поле может содержать имя команды шрифтового оформления без обратного слэша, которая будет использована для вывода номера страницы. Например, номер страницы элемента `\index{лазеры|textbf}` будет жирным. Отметим, что в стандартных настройках xindy для оформления номера доступны только жирный шрифт и курсив.

Покажем, как можно изменить шрифты, чтобы сделать более наглядным приведенный выше пример:

Предметный указатель

Л

<code>\index{лазеры textbf}</code>	↔	лазеры, 12, 39, 115
<code>\index{лазеры!\textit{газовые}}</code>	↔	<i>газовые</i> , 87
		CO ₂ , 96
		He-Ne, 89
		<i>твердотельные</i> , 15

Круглые скобки в поле управления выводом ссылки, т. е. |(и |), позволяют указать диапазон страниц. Для этого используются две команды. Первая (`\index{описание элемента|}`) ставится в начале диапазона, а вторая (`\index{описание элемента|}`) в его конце, при этом описания должны быть идентичны. Шрифт страниц диапазона также можно регулировать, поставив после |(и |) имя команды шрифтового выделения. Например, `\index{описание элемента|textbf}`.

Вместо номера страницы в ссылке можно внести комментарий:

```
\index{описание записи|see{текст}}
```

```
\index{описание записи|seealso{текст}}.
```

В первом случае перед текстом будет добавлено «см.», а во втором — «см. также». Комментарий является обычным текстом L^AT_EX. Выводимый перед ним текст, хранящийся в командах

```
\seename
```

и

```
\seealsoname,
```

можно изменить с помощью `\renewcommand`.

Подводя итог описания синтаксиса команды `\index`, отметим, что символы !@| служат для разделения полей ее записей. Если же они используются как обычные символы, их следует защитить двойной кавычкой, например `\index{"! --- факториал}`. То же относится и к самой кавычке: `\index{"" --- двойная кавычка}`.

С точки зрения верстки, команды `\index` являются пробелами, поэтому, во избежание недоразумений, их нужно ставить *после* знаков препинания, скобок и т. д.

11.1. Верстка указателей

Рассмотрим теперь методы тонкой настройки формирования и вывода указателя. Пакет `imakeidx` [35] обеспечивает составление сразу нескольких указателей, каждому из которых можно присвоить имя, дающее доступ к индивидуальной настройке параметров сортировки и верстки. Данные, используемые для генерации указателей, перед сортировкой должны быть разнесены по разным файлам; `imakeidx` делает это непосредственно во время компиляции, открывая для каждого указателя индивидуальный файл записи. Ввиду ограниченности числа таких файлов предусмотрена также возможность сохранения одного общего файла с его последующим разделением на компоненты. Используя Perl-скрипт `splitindex` для проведения этой операции, пакет `imakeidx` способен наладить автоматическое разделение и сортировку любого числа указателей.

Действия пакета настраиваются следующими параметрами, перечисляемыми через запятую при загрузке пакета.

`makeindex` | `texindy` | `xindy` | `truexindy` — имя программы, вызываемой для автоматической сортировки указателей. По умолчанию считается, что ей является `MakeIndex`. Синонимы `texindy` и `xindy` указывают, что сортировку будет делать `texindy`. Параметр `truexindy` используется, если программа `xindy` загружается с собственными настройками, отличными от настроек `texindy`.

`splitindex` — автоматический запуск скрипта, расщепляющего файл с данными для указателей на компоненты.

`quiet` — подавление вывода сообщений автоматически запускаемых программ.

`noautomatic` — отключение автоматического вызова скрипта `splitindex` и программ сортировки.

`nonewpage` — подавление вывода команды `\newpage`, которая по умолчанию ставится перед каждым указателем.

`original` — использование окружения `theindex`, определенного классом компилируемого документа, без дополнений пакета `imakeidx`.

Одно из множества достоинств пакета `imakeidx` заключается в индивидуальных настройках генерации каждого указателя, доступ к которым предоставляет команда

```
\makeindex[список настроек] .
```

Приведенные ниже настройки перечисляются через запятую в ее параметре.

`name` = имя указателя — присваивает указателю имя, используемое для генерации файлов и адресации других команд. Если имя отсутствует, настройки применяются к указателю, формируемому по умолчанию, файлы которого наследуют имя компилируемого документа.

`title` = {заголовок указателя} — устанавливает заголовок указателя. Если параметр отсутствует, используется заголовок, хранящийся в команде `\indexname`.

`program` = `makeindex` | `texindy` | `xindy` | `truexindy` — устанавливает программу, вызываемую для сортировки указателя, если она отличается от указанной при загрузке пакета `imakeidx`.

`options` = список ключей — устанавливает список ключей, передаваемый программе сортировки (см. приложение Г), например, `options= -C utf8 -L russian -M rueng` .

`noautomatic` — подавляет автоматический вызов программы сортировки, используемый по умолчанию.

`intoc` — вносит указатель в оглавление.

`columns` = число — задает количество колонок в указателе, по умолчанию равное двум.

`columnsep` = длина — задает размер интервала между колонками, по умолчанию для двух колонок равный 35pt.

`columnseprule` — разделяет колонки вертикальной линией.

Команды `\makeindex[•]` должны находиться в преамбуле документа. Приведем пример команды, формирующей англо-русский указатель с именем `pkg` и заголовком `Пакеты`:

```
\makeindex[name= pkg, title= Пакеты,%  
options= -C utf8 -L russian -M rueng -M hyperref]
```

Предполагается, что при загрузке `imakeidx` сортирующей программой объявлена `texindy`. Ключи, настраивающие сортировку, описаны в прил. Г. Первый из них объявляет кодировку сортируемого файла, второй загружает настройки русского языка, третий обеспечивает англо-русскую сортировку и, наконец, последний оформляет номера страниц указателя в виде гиперссылок.

Составление и вывод нескольких указателей обеспечивают команды

```
\index[имя указателя]{описание записи},  
\printindex[имя указателя].
```

Указываемое в параметре `имя` связывает их с определенным указателем. Команды без параметра адресуются к указателю, формируемому по умолчанию.

Каждый указатель можно снабдить введением, которое выводится после заголовка, предваряя список элементов. Для этого определена команда

```
\indexprologue[вертикальный пробел]{текст}.
```

Ее аргументом является текст введения, а в параметре можно задать пробел, отделяющий введение от элементов указателя. По умолчанию используется пробел `\bigskip`.

Вставив введение, команда `\printindex` аннулирует его, чтобы оно не появилось в следующем указателе, поэтому команду `\indexprologue` при необходимости следует «обновлять» перед каждым выводом указателя.

Упомянувшийся выше указатель с именем `pkg`, представленный на с. 344, составлен с помощью команд

```
\index[pkg]{название пакета}
```

и вставлен в документ следующим образом:

```
\indexprologue{
  \label{p:Packages}
  Символы \texttt{[•]} указывают на наличие настроек.}
\printindex[pkg]
```

Метка, поставленная во введении, помогла сослаться на страницу с указателем.

Общую настройку оформления указателей делает команда

```
\indexsetup[список настроек],
```

которую лучше разместить в преамбуле вместе с командами `\makeindex`. Как обычно, настройки перечисляются через запятую в ее параметре.

`headers = {левый колонтитул}{правый колонтитул}` — установка колонтитулов для страниц с указателями.

`firstpagestyle = стиль страницы` — установка стиля первой страницы указателей. По умолчанию используется `plain`.

`noclearpage` — подавление команды `\clearpage`, по умолчанию выполняемой между указателями.

`toclevel = имя команды разбивки` — уровень указателей в оглавлении, например `toclevel = section`. По умолчанию указатели являются главами.

`level = команда разбивки` — стиль вывода заголовков указателей. По умолчанию используется `\chapter*`. Команда с

именем без звездочки автоматически вставляет указатели в оглавление. В этом случае не следует использовать параметр `intoc` в командах `\makeindex`.

`othercode` = набор команд — дополнительные команды, выполняемые перед выводом элементов указателей, обычно применяемые для изменения параметров верстки. Например, `othercode=\small` уменьшит шрифт всех указателей.

Последняя из команд, введенных пакетом `imakeidx`, позволяет сформировать командную строку скрипта `splitindex`:

```
\splitindexoptions{список ключей},
```

который по умолчанию запускается без ключей. Ее также следует поместить в преамбуле.

Подводя итог обзора команд, перечислим основные настройки верстки указателя. Команда `\indexsetup` позволяет сформировать колонтитулы (`headers`), установить стиль оформления элементов указателя (`overcode`) и заголовка (`level`), а также задать его уровень в оглавлении (`toclevel`). Команда `\makeindex` вносит указатель в оглавление (`intoc`), устанавливает его заголовок (`title`), число колонок (`columns`) и расстояние между ними (`columnsep`). Команда `\indexprologue` позволяет сформировать для указателя введение.

Пакет `imakeidx` может автоматически вызывать программы, необходимые для составления указателей, непосредственно во время компиляции. Это не требуется, если формируется один указатель, так как оболочка `TeXstudio` способна сама настроить и запустить программу сортировки по окончании компиляции. Проблема возникает при генерации нескольких указателей, в которой может участвовать несколько программ с разными настройками. В этом случае помощь пакета `imakeidx` очень существенна, но она реализуется только если компилятор предоставляет ему возможность запуска внешних программ, обычно отключаемую в целях безопасности. Для активации этой возможности к строке запуска компиляторов следует добавить ключ `-shell-escape`, es-

ли используется дистрибутив TeXLive, или `-enable-write18` для дистрибутива MikTeX. В программе TeXstudio командную строку `pdflatex` можно отредактировать в меню «Options ⇒ Configure TeXstudio. . . ⇒ Commands ⇒ PdfLaTeX».

Заключительные замечания касаются мнемоники имен файлов, генерируемых при формировании указателей. Если скрипт `splitindex` не используется, файлы получают имя указателя и расширения `idx` и `ind`, а файлы указателя, формируемого по умолчанию (без имени), наследуют имя компилируемого документа. При использовании скрипта `splitindex` файл с данными всех указателей получает имя документа, а имена компонент составляются из имен документа и указателей, разделенных дефисом. Например, когда компилируется файл `document.tex` и создаются три указателя с именами `text`, `math` и без имени, то в первом случае генерируются файлы `document.●`, `text.●` и `math.●`, а во втором — файл `document.idx`, который затем разделяется на `document-text.●`, `document-math.●` и `document-document.●`. Здесь ● означает расширения `idx` и `ind`.

11.2. Глоссарий

Проще всего список терминов, т. е. *глоссарий*, или список сокращений — *аббревиатур*, сверстать в окружении `description`, но можно также воспользоваться и механизмом формирования указателей. Помимо команды `\index{●}` в L^AT_EX определена команда

```
\glossary{описание записи} ,
```

имеющая такой же синтаксис. Пакет `makeidx` обеспечивает обработку этих команд, если она инициирована в преамбуле документа командой

```
\makeglossary .
```

В этом случае при компиляции они преобразуются в команды

```
\glossaryentry{описание элемента}{номер страницы}
```


и направляются в файл с расширением `•.glo`. В ядре ЛАТ_EX отсутствуют средства генерации глоссария и списка аббревиатур, но можно воспользоваться программой `texindy` со специальными настройками, описанными в прил. Г.

Программа `texindy` генерирует глоссарий в виде окружения `theindex`, сохраняя его в отдельный файл (см. прил. Г). Команда аналогичная команде `\printindex`, вставляющей в документ указатель, для глоссария не предусмотрена, поэтому придется делать это вручную. Загрузку глоссария, находящегося в файле `glossary.tex`, можно оформить следующим образом:

```
\renewcommand\indexname{Глоссарий}  
\include{glossary},
```

а если файл имеет расширение, отличное от `•.tex`, нужно использовать загрузку `\input`.

Стандартный указатель разбивается по алфавиту на рубрики, и в начале каждой рубрики выводится большая буква-заголовок, что вряд ли требуется в глоссарии. Список терминов или аббревиатур не должен отсылать читателя к страницам, на которых они встречаются, как это делают элементы указателя. Настройки `texindy` подавляют их вывод заголовков и номеров страниц.

Объемный текст расшифровки термина плохо смотрится при верстке в несколько колонок, применяемой для вывода указателя. Если формат одной колонки более предпочтителен для глоссария, можно воспользоваться пакетом `imakeidx`, который не только обеспечит его, но и существенно упростит процедуру создания глоссария.

Сформируем глоссарий в виде «указателя» с именем `glo`. Для его генерации определим в преамбуле команду

```
\makeindex[name= glo, columns= 1,%  
           title= Элементарные частицы,%  
           options= -C utf8 -L russian -M rueng -M glossary]
```

Назначение первых трех ключей объяснено ранее, а последний загружает настройки, необходимые для генерации глоссария.

Описания терминов нужно поместить в команды
или `\index[glo]{\textsf{термин} — описание}`
`\index[glo]{Рубрика!\textsf{термин} — описание}`.

Первая из них заносит термин в общий список глоссария, а вторая — в определенную его рубрику. Синтаксис команд `\index` и `\glossary` поддерживает три уровня записей, которые можно использовать, чтобы сформировать рубрики. Термины каждой рубрики сортируются отдельно.

Обратите внимание на изменение шрифта термина. Программа `texindy` не может выделить его из сортируемого текста, поэтому об оформлении термина должен позаботиться сам автор.

Глоссарий, сформированный программой `texindy`, вставляется в документ как обычный «указатель» командами

```
\indexprologue  
\printindex[glo].
```

Следует учесть, печать указателя прекращает дальнейший сбор его элементов, поэтому команды `\index`, находящиеся после команд `\printindex`, не обрабатываются. Для верстки предметного указателя, располагающегося в конце документа, это не создает проблем, однако глоссарий может находиться в его начале. В таком случае на первом этапе, чтобы собрать информацию, следует закомментировать команду `\printindex`. Затем, раскомментировав ее, определить количество страниц, занимаемых глоссарием, вновь закомментировать ее и сдвинуть счетчик страниц на полученное значение, чтобы установить правильную нумерацию в указателях. На конечном этапе следует раскомментировать печать глоссария, убрать сдвиг страниц и исключить обновление указателей, не запуская более формирующую их программу.

Разберем верстку глоссария, показанного на рис. 11.2. Командой `\makeindex[glo]{•}` заданы верстка в одну колонку и заголовок. Введение создано командой `\indexprologue` при загрузке глоссария. Для остального анализа приведем часть записей, которые в программе могут быть рассеяны в любом порядке, а здесь собраны в логическую цепочку, позволяющую понять конструкцию в целом:

Элементарные частицы

Элементарные частицы — первичные, далее неразложимые частицы, из которых, предположительно, состоит вся материя. Они подчиняются квантовой статистике и с этих позиций делятся на *бозоны* и *фермионы*.

Бозоны

Бозоны — частицы с нулевым или целым спином.

Мезон — нестабильная частица, принадлежащая классу адронов.

Пи-мезоны — группа нестабильных сильно взаимодействующих нейтральных (π^0), положительно (π^+) или отрицательно (π^-) заряженных частиц, обладающих массой, промежуточной между массами *протона* и *электрона*.

Спин 0.

Фотон — квант электромагнитного излучения. Спин 1.

Фермионы

Фермионы — частицы с полуцелым спином.

Нейтрино — стабильная легкая (возможно безмассовая) незаряженная частица. Спин $1/2$.

Нейтрон — незаряженная частица с большим временем жизни. Спин $1/2$.

Позитрон — стабильная положительно заряженная частица, античастица *электрона*. Спин $1/2$.

Протон — стабильная положительно заряженная частица. Спин $1/2$.

Электрон — стабильная отрицательно заряженная частица. Спин $1/2$.

Рис. 11.2. Пример глоссария

```

\index[glo]{"!@\normalsize}...
\index[glo]{Б@\textbf{Бозоны}}...
\index[glo]{Бозон@\textsf{бозоны} --- частицы с...}...
\index[glo]{Бозон!Мезон@\textsf{мезон} --- нестабильн...}
\index[glo]{Бозон!Мезон!\textsf{пи-мезоны} --- группа...}
\index[glo]{Бозон!\textsf{фотон} --- квант...}...

```

Прежде всего нужно настроить размер шрифта глоссария. Обычно для указателей устанавливается мелкий шрифт, например `\indexsetup{othercode=\small}`, который используется для всех окружений `theindex`. Шрифт введения не связан со шрифтом указателя, поэтому его изменение в команде `\indexprologue` на текст указателя не повлияет. Настроить шрифт отдельного указателя внешними средствами не удастся и нужно действовать «изнутри».

Первая команда `\index` содержит декларацию `\normalsize`, устанавливающую стандартный размер шрифта. Область ее действия распространится на весь текст, содержащийся в окружении `theindex`, если запись после сортировки окажется первой. Заголовков записи `!`, используемый для сортировки, состоит из восклицательного знака, защищенного двойной кавычкой, который является первым неслужебным символом таблицы `unicode`, поэтому `texindy` поставит данную запись впереди всех остальных.

Таким образом, первая команда `\index` устанавливает размер шрифта глоссария. Остальные формируют рубрику глоссария **Бозоны**, название которой выводит вторая команда.

Заголовки записей второй и третьей команд начинаются с одинаковой буквы, но чтобы после сортировки вторая запись оказалась впереди, ее заголовок сокращен до одной буквы «Б».

Третья команда расшифровывает термин **бозоны**, а также вводит заголовок записей «Бозон», по которому отбираются остальные термины данной рубрики.

Четвертая команда выводит описание термина **мезон** и организует вложенную рубрику, вводя для ее сортировки заголовки «Бозон!Мезон».

Пятая команда выводит определение термина **пи-мезоны** на третьем уровне глоссария.

Наконец, шестая команда добавляет определение **фотона** ко второму уровню глоссария.

Рубрика **Фермионы** организована аналогичным образом.

Рубрики **Бозоны** и **Фермионы** разделены небольшим вертикальным пробелом, вставленным командой `\indexspace`. В окружении `theindex` он разделяет рубрики первого уровня. В данном случае пробел смотрится органично, однако список аббревиатур, состоящий только из терминов первого уровня, может выглядеть «рыхлым», тогда дополнительный пробел лучше убрать. Чтобы это сделать, перед выводом глоссария нужно переопределить команду `\indexspace` следующим образом:

```
или
      { \let\indexspace\relax
        \printindex[•] }
      { \let\indexspace\relax
        \include{•} }
```

Команда `\relax` означает «пропустить операцию», а команда `\let` присваивает ее значение команде `\indexspace`. Как говорилось выше, такой метод присваивания определен непосредственно в **TEX**. Обратите внимание на фигурные скобки, охватывающие команды присваивания и загрузки, они необходимы, чтобы при выходе из созданной ими группы ввод пробела восстановился.

Отметим, что представленная в прил. Г настройка `texindy` позволяет изъять дополнительный пробел глобально, но возможность регулировки его появления перед загрузкой глоссария представляется более оптимальной, так как позволяет выбрать предпочтительный вариант оформления глоссария в каждом конкретном случае.

Используя пакет `imakeidx`, можно сверстать несколько глоссариев и списков аббревиатур, например свой «комплект» для каждой части или главы книги. Еще более богатые возможности предоставляет пакет `glossaries`. Он позволяет настроить все аспек-

ты верстки таких списков и вводит для этого несколько десятков команд, подробно описанных на трехстах пятидесяти страницах его документации [36]. Заинтересованный читатель может ознакомиться с его ресурсами из краткого введения, содержащего «всего» тридцать семь страниц [37].

Глава 12

Стандартные пакеты

Возможности верстки зависят от пакетов, загружаемых при компиляции документа. В предисловии отмечалась опасность использования пакетов сторонних разработчиков. Как правило, при компиляции в издательстве их загрузка блокируется, так как они могут изменить параметры верстки или привести остановке компиляции из-за несовместимости с другими пакетами, необходимыми для верстки тома журнала. В то же время стандартные пакеты зачастую не только не блокируются, но и рекомендуются к использованию.

Готовя рукопись к публикации, авторы должны пользоваться только пакетами, рекомендованными издателем, однако верстая «собственные» документы, можно подключать и дополнительные, но и при этом тоже стоит ограничиться стандартными, большая часть из которых уже обсуждалась. В частности, ранее упоминалась коллекция пакетов `extsizes`, определяющая размер шрифта `14pt`, требуемый для оформления пояснительных записок к диплому, диссертаций и авторефератов, для верстки которых можно использовать класс `extreport` из этой коллекции, см. разд. 2.6.2.

Стандартной является коллекция пакетов `tools`, поддерживаемая и обновляемая по мере необходимости разработчиками ядра \LaTeX : www.ctan.org/pkg/latex-tools. Добавляемые ими

новые возможности, команды и окружения подробно описаны в книгах [3, 4] и их документации. Значительная часть пакетов этой коллекции уже обсуждалась ранее, к ним относятся `afterpage`, `array`, `calc`, `dcolumn`, `indentfirst`, `longtable`, `tabularx` и `theorem`. Среди оставшихся перечислим лишь пакеты, имеющие отношение к верстке, и укажем их назначение.

12.1. Пакеты коллекции `tools`

Для печати жирных символов в формулах пакет `bm` [38] вводит команду `\bm{•}`, аналогичную обсуждавшейся выше команде `\boldsymbol{•}`.

Пакет `delarray` [39] позволяет заключить таблицы `tabular` и `array` в скобки.

Пакет `enumerate` [40] определяет параметр для списка `enumerate`, настраивающий формат вывода номеров записей.

Пакет `ftnright` [41] при многоколоночном наборе помещает сноски `\footnote{•}` в правую колонку.

Пакет `multicol` [42] вводит новое окружение `multicols`, позволяющее на одной странице совмещать одно- и многоколоночный набор и верстать до десяти колонок текста.

Пакет `varioref` [43] расширяет возможности перекрестного цитирования (см. с. 32). Он вводит ряд новых команд, позволяющих очень гибко настраивать форматы вывода ссылок на рисунки, таблицы, номера страниц и т. д.

Пакет `verbatim` [44] снимает ограничения на объем текста, выводимого окружениями `verbatim` и `verbatim*` (см. с. 64). Для обширных комментариев он определяет окружение `comment`.

Пакет `xspace` [45] вводит одноименную команду `\xspace`, вставляющую пробел, если только за ней не следует знак препинания, одна из фигурных скобок, закрывающая круглая скобка, прямой или обратный слэш.

12.2. Условные конструкции пакета `ifthen`

Набор низкоуровневых команд позволяет управлять действиями компилятора `TeX` в зависимости от выполнения различных условий. Пакет `ifthen` [47] коллекции `base`, в которую также входят все стандартные классы, определяет для `LaTeX` две условных конструкции:

```
\ifthenelse{условие}{блок да}{блок нет} ,  
\whiledo{условие}{тело цикла} .
```

Обе команды проверяют условие, значениями которого являются логические единицы `true` (верно) и `false` (неверно). Если оно верно, команда `\ifthenelse` выполняет блок `да`, а в противном случае — блок `нет`. Команда `\whiledo` при верном условии выполняет тело цикла и вновь возвращается к его проверке до тех пор, пока условие станет неверным.

Элементарными проверками являются:

`=`, `<`, `>` — сравнение целых чисел, например `\value{•} < 5`;

`\isodd{число}` — определение четности, дающее значение `true` для нечетных чисел;

`\lengthtest{длина1 • длина2}` — сравнение жестких длин (символ • означает операторы `=`, `<`, `>`);

`\equal{строка1}{строка2}` — сравнение строк;

`\boolean{переменная}` — определение значения логической переменной.

Из элементарных проверок с помощью логических операций

`\and` (и), `\or` (или), `\not` (отрицание)

и группирующих команд-скобок `\(` и `\)` можно построить комплексное условие, например проверку того, что текущая страница имеет четный номер, а ее лист — альбомную ориентацию:

```
\whiledo{... \and\ ( \lengthtest{\paperheight < \paperwidth}
  \and \not\isodd{\value{page}} \)}{ тело цикла }.
```

Опущенная часть первой операции `\and` обозначена точками.

В проверке эквивалентности строк используется их внутреннее представление, которое у двух строк, выглядящих при печати одинаково, может различаться. Например, отрицательный результат даст проверка `\equal{\IeC{\cyra}}{a}`, так как буква «а» и ее командное представление (`\IeC{\cyra} \rightsquigarrow a`) не эквивалентны.

Логическую переменную создают декларации

```
\newboolean{имя},
\provideboolean{имя}.
```

Вторая из них делает это, только если такой переменной еще нет. Переменная не является командой, поэтому ее имя не должно содержать символ «\».

Значение переменной устанавливает команда

```
\setboolean{имя}{значение}.
```

Указав `true` или `false`, значение можно задать явно, но можно установить его и с помощью условия. Результат его анализа будет присвоен переменной. Например, команда

```
\setboolean{pagerange}{ \value{page} > 49
  \and \value{page} < 101 }
```

задаст переменной `pagerange` значение `true` в диапазоне страниц от пятидесяти до ста и значение `false` для остальных страниц.

В ядре \LaTeX определена еще одна условная конструкция:

```
\IfFileExists{имя файла}{блок да}{блок нет}ifthen ,
```

проверяющая наличие файла. Если он найден, выполняется блок `да`, а если отсутствует — блок `нет`. Файл ищется не только в рабочей папке, но и во всех директориях, доступных компилятору с помощью средств поиска, описанных в прил. А.

12.3. Пакет `epstopdf`

В разд. 3.1.1 говорилось, что программа `epstopdf` конвертирует `eps`-рисунок в формат `pdf`. Стандартный пакет `epstopdf` [48] из коллекции `epstopdf-pkg` позволяет делать это непосредственно во время компиляции. Для этого пакет `graphicx` нужно загрузить с параметром `pdftex`, дополнительно загрузив сам пакет `epstopdf`. Кроме того, чтобы иметь возможность взаимодействовать с конвертирующей программой, в командной строке компиляторов `LaTeX` нужно добавить ключ `-shell-escape`, если используется дистрибутив `TeXLive`, или `-enable-write18` для дистрибутива `MikTeX`.

Когда все условия выполнены, команды `\includegraphics` для файлов с расширением `•.eps` ищут соответствующий `•.pdf` файл, а если его нет, вызывают программу `epstopdf` и затем загружают конвертированный файл.

Действия пакета `epstopdf` контролируют параметры, устанавливаемые при его загрузке, или после нее. Для настройки предназначена команда

```
\epstopdfsetup{список параметров} ,
```

в аргументе которой через запятую перечисляются параметры. Приведем лишь те из них, что используются наиболее часто.

`update` — конвертировать файл только, если `pdf`-файл не существует, или создан позже, чем `eps`-файл.

`suffix=•` — добавить суффикс `•` к имени `pdf`-файла. При установке параметра `suffix=-generated` файл `sample.eps` конвертируется в `sample-generated.pdf`.

`prefersuffix= true | false` — вставлять в документ файлы с суффиксом (`true`) или без него (`false`).

Последний параметр определяет выбор файлов, если в рабочей папке находятся два сорта `pdf`-файлов, среди которых одни имеют

те же имена, что и исходные eps-файлы, а имена вторых содержат суффикс, указанный параметром `suffix`.

Параметры `update`, `suffix=-eps-converted-to`, `prefersuffix=true` заданы по умолчанию, т. е. генерируются и загружаются файлы с именами `•-eps-converted-to.pdf`.

По умолчанию конвертирует файлы программа `epstopdf`, но настройки позволяют использовать и другие программы. Также можно настроить конвертацию файлов с форматами, отличными от `eps` [48].

12.4. Пакеты `lscapе` и `pdfscape`

Пакет `lscapе` из коллекции `latex-graphics` [13] вводит окружение `landscape`, формирующее страницу с альбомной ориентацией. Перед изменением формата страницы выполняется команда `\clearpage`, завершающая текущую страницу и выводорящая «находящиеся в очереди» плавающие объекты. Ориентация колонтитулов на странице, верстаемой окружением `landscape`, не меняется.

Пакет `pdfscape` [49] обеспечивает поддержку окружения `landscape` в pdf-документах. Он автоматически загружает пакет `lscapе` и поворачивает страницу на девяносто градусов, добавляя к ее параметрам атрибут `/Rotate`.

12.5. Пакет `microtype`

Строго говоря, пакет `microtype` [50] к числу стандартных не относится, но он не вносит в текст никаких команд, поэтому при его использовании манускрипт остается стандартным. Если стандартные алгоритмы `TeX` оптимизируют заполненность строк, распределяя в них слова, то пакет `microtype` идет дальше, он слегка меняет ширину букв в словах, корректируя символы шрифтов. Для таких операций лучше всего подходит pdf-формат

документа и векторные шрифты, например упоминавшиеся в разд. 1.1 шрифты `cm-super`.

Хотя пакет `microtype` имеет параметры, его можно загружать и без них, так как настройки по умолчанию хорошо отлажены, и их лучше не трогать. Загрузив пакет, с помощью команды

```
\UseMicrotypeSet [операции] {шрифты}
```

следует выбрать один из готовых наборов операций и шрифтов. Наиболее полным является набор

```
\UseMicrotypeSet [protrusion]{basicmath},
```

оптимизирующий расстояния между словами и ширины букв всех текстовых шрифтов нормальной насыщенности, исключая моноширинные, в диапазоне размеров от `\footnotesize` до `\large` (см. табл. 2.6). Вдобавок к этому оптимизируются ширины букв математических шрифтов с кодировками `OML` и `OMS` [3].

Пакет `microtype` вводит и другие команды, позволяющие настраивать различные операции и сформировать свои собственные наборы операций и шрифтов, но лучше довериться профессионалам, разработавшим настройки, используемые по умолчанию. Шрифты — очень тонкий инструмент, к которому нужно относиться с большим уважением и осторожностью.

12.6. Гиперссылки пакета `hyperref`

В электронном документе ссылки на литературу, формулы, рисунки и т. д. удобно преобразовать в гиперссылки, позволяющие легко перейти к объекту ссылки. Такую возможность предоставляет пакет `hyperref`. Он также способен настроить практически все аспекты, связанные с представлением pdf-документа, вплоть до деталей оформления окна открывающей его программы и корректировки пунктов ее меню. Пакет имеет чрезвычайно большое число настроек, вводит новые команды и переопределяет большое количество стандартных, поэтому его рекомендуется загружать последним. Если он добавляется в программу, которая уже компи-

лировалась, рекомендуется удалить все промежуточные файлы, иначе в первые два прохода компиляции могут появиться ошибки. То же нужно сделать и после его отключения. Не обсуждая богатые ресурсы пакета `hyperref`, описанные в его документации [46] и книгах [3, 5], упомянем лишь минимальный набор настроек и команд, относящихся непосредственно к гиперссылкам.

При загрузке пакета `hyperref` ссылки, генерируемые командами `\ref{•}`, `\pageref{•}`, `\eqref{•}` и `\cite{•}`, автоматически становятся гиперссылками, а чтобы сделать обычную ссылку введены команды

`\ref*{метка}` и `\pageref*{метка}`.

Вдобавок к ним определена команда

`\autoref{метка}`,

которая в зависимости от типа объекта, на который ссылаются, добавляет перед номером контекстную вставку, например номер формулы `\autoref` выведет в виде «Equation •» в английском тексте и «выр. •» в русском. Контекстные вставки хранятся в командах, перечисленных в табл. 12.1. С помощью переопределения `\renewcommand` их можно изменить.

В дополнение к стандартным ссылкам определена команда

`\hyperref{имя метки}{текст}`.

Ее первым аргументом является имя метки, заданной командой `\label`, а во второй аргумент помещается текст, оформляемый в виде гиперссылки. Этим она отличается от команд `\ref`, `\pageref`, `\eqref` и `\cite`, у которых гиперссылками становятся выводимые номера или ярлыки.

Команда `\hyperref` адресуетя к объектам, находящимся в самом документе, а чтобы делать гиперссылки к внешним ресурсам, определена команда:

`\href{ресурс}{текст}`.

Ее второй аргумент также задает текст, оформляемый в виде гиперссылки. Ресурс может быть интернет-адресом, адресом элек-

Таблица 12.1

Контекстные вставки команды `\autoref`

Команды	Контекстные вставки	
<code>\AMSautorefname</code>	Equation	выр.
<code>\equationautorefname</code>	Equation	выр.
<code>\theoremautorefname</code>	Theorem	теор.
<code>\Hfootnoteautorefname</code>	footnote	подстр. прим.
<code>\figureautorefname</code>	Figure	рис.
<code>\Itemautorefname</code>	item	п.
<code>\pageautorefname</code>	page	с.
<code>\tableautorefname</code>	Table	табл.
<code>\appendixautorefname</code>	Appendix	прил.
<code>\chapterautorefname</code>	chapter	гл.
<code>\paragraphautorefname</code>	paragraph	п.
<code>\partautorefname</code>	Part	ч.
<code>\sectionautorefname</code>	section	разд.
<code>\subsectionautorefname</code>	subsection	разд.
<code>\subsubsectionautorefname</code>	subsubsection	разд.

тронной почты или путем к файлу. Переход по гиперссылке ведет к выполнению различных действий, зависящих от идентификатора ресурса.

Адрес почты должен начинаться с идентификатора `mailto:..`. Например при переходе по ссылке `e-mail`, созданной командой

```
\href{mailto:anybody@gmail.com}{e-mail},
```

откроется окно почтовой программы для редактирования письма с заготовленным адресатом.

Путь к файлу должен начинаться либо с идентификатора `file://` или `run:`, либо со слэша или точки.¹ Если идентифи-

¹ Путь к файлу указывается в Unix нотациях, т. е. папки разделяются символом `/`, в относительном пути рабочая папка обозначается точкой, а родительская — двумя точками. В операционной системе Windows абсолютный путь начинается с идентификатора диска, например `c:/Program Files/...`

катор отсутствует, автоматически используется `file://`. Например, команда `\href{run:sample.avi}{видео}` создаст гиперссылку **видео** для запуска медиафайла, находящегося в той же папке что и pdf-документ. Команда

```
\href{d:/Photos/sample.jpg}{фотография}
```

привяжет гиперссылку **фотография** к снимку `sample.jpg` в папке `Photos` на диске `d:`.

Результат действий, совершаемых при переходе по гиперссылке, зависит от типа файла, настроек программы, показывающей pdf-документ, и настроек безопасности операционной системы. При правильных настройках файл должен открываться программой, ассоциированной в операционной системе с его расширением. Для ссылок, начинающихся с ключевого слова «`run:`», программа `AcrobatReader` использует ассоциации операционной системы, а в остальных случаях действуют ее собственные ассоциации.

Интернет-адрес обычно начинается с идентификатора типа ресурса `http://`, `https://`, `ftp://` и т. д. Например:

```
\href{https://en.wikibooks.org/wiki/LaTeX}{LaTeX},  
\href{www.google.com}{google}.
```

Если идентификатор не указан, автоматически используется `http://`. Переход по гиперссылке запускает браузер, загружающий нужную интернет-страницу.

Для ссылок на интернет-ресурсы дополнительно введена команда:

```
\url{интернет-ресурс}.
```

Ее аргумент используется и как адрес, и как текст гиперссылки. Аналог данной команды

```
\nolinkurl{интернет-адрес}
```

вставляет в документ только адрес, не создавая гиперссылку.

Для создания гиперссылок пакет `hyperref` использует упоминающуюся в разд. 6.4 команду `\refstepcounter`, связывающую метки со значениями счетчиков. Но так как в нумерации страниц она не участвует, гиперссылка, формируемая командой `\pageref`, может указывать не на ту страницу, хотя ее номер и будет указан

правильно. Чтобы этого не происходило, пакет `hyperref` вводит специальный счетчик и команду

```
\phantomsection,
```

связывающую его с метками. Ее нужно ставить перед командами `\label`, чтобы пометить страницы, на которые предполагается ссылаться, например:

```
\phantomsection\label{p:sample}    ...  
\pageref{p:sample}.
```

Эту команду можно использовать также, если гиперссылка элемента оглавления, вносимого в вручную, ведет к неверной странице:

```
\phantomsection  
\addcontentsline{toc}{chapter}{Указатель фамилий}.
```

Рассмотрим теперь вкратце настройку пакета `hyperref`. Ее можно сделать, используя параметр команды `\usepackage` при загрузке пакета, перечислив настройки списком через запятую, например

```
\usepackage[colorlinks,bookmarks,unicode]{hyperref}.
```

Многие настройки имеют значение `true` (используемый), `false` (неиспользуемый) или присваивают цвет. Значение можно установить путем присваивания, например `hyperindex = false`, а если параметр указан без присваивания, используется `true`.

Формат присваивания цвета зависит от оформления гиперссылки. Для рамки, рисуемой вокруг нее, указываются значения компонент палитры `rgb`, разделенные пробелом, а для текста гиперссылки — имя цвета, например

```
citebordercolor = .501, citecolor = blue.
```

Задание цвета текста палитрой не поддерживается, поэтому имена нестандартных цветов нужно определять заранее.

Перечислим основные настройки и их функции.

`bookmarks` — формирует электронное оглавление, служащее для быстрого перемещения по документу, которое обычно по-

является слева от текста в окне программ просмотра pdf-документов.

`unicode` — обеспечивает unicode-кодировку pdf-конструкций, в частности электронного оглавления.

`hyperindex` — формирует гиперссылки предметного указателя.

`linktocpage` — оформляет в виде гиперссылок номера страниц в оглавлении. По умолчанию гиперссылками становятся названия разделов.

`colorlinks` — выделяет цветом текст гиперссылки. По умолчанию вокруг гиперссылки рисуется тонкая цветная рамка, а ее текст цветом не выделяется.

`pdfborder` — устанавливает толщину рамок, окружающих гиперссылки. По умолчанию ее значение равно `0 0 1`, где единица означает `1pt`. При использовании параметра `colorlinks` значение устанавливается в `0 0 0`.

`allbordercolors` — устанавливает цвет для всех рамок. По умолчанию не используется.

`allcolors` — устанавливает цвет всех гиперссылок. По умолчанию не используется.

`anchcolor` — устанавливает цвет текста, служащего объектом гиперссылки. По умолчанию используется черный (`black`). Данную настройку `pdflatex` не поддерживает.

`citecolor`, `citebordercolor` — устанавливает цвет гиперссылок на библиографию. По умолчанию используется зеленый (`green`, `0 1 0`).

`filecolor`, `filebordercolor` — устанавливает цвет гиперссылок на файлы. По умолчанию используется голубой (`cyan`, `0 .5 .5`).

`linkcolor`, `linkbordercolor` — устанавливает цвет гиперссылок на объекты, помеченные командами `\label`, а также для сносков, пунктов оглавления и страниц указателя. По умолчанию используется красный (`red`, `1 0 0`).

`menucolor` — устанавливает цвет пунктов меню программы просмотра pdf-документов, изменяемых самим документом. По умолчанию используется красный (`red`, `1 0 0`).

`runcolor`, `runbordercolor` — устанавливает цвет гиперссылок с идентификатором `run:`, запускающих внешние приложения. По умолчанию для рамки используется цвет `0 .7 .7`, а для текста — цвет `filecolor`.

`urlcolor`, `urlbordercolor` — устанавливает цвет гиперссылок на интернет-ресурсы. По умолчанию используется пурпурный (`magenta`, `0 1 1`).

Отметим, что параметр `unicode` необходим для корректной русификации электронного оглавления документа.

Изменить настройки можно и после загрузки пакета `hyperref`. Для этого определена команда

```
\hypersetup{список настроек} .
```

Список настроек в ее аргументе перечисляется через запятую:
`\hypersetup{bookmarks=true, unicode=true,
colorlinks=true, citecolor=blue, urlcolor=cyan}`.

Хотя команду `\hypersetup` можно использовать многократно как в преамбуле, так и в любой другой части программы, часть параметров устанавливается только в преамбуле. Например, вне преамбулы изменить стиль оформления гиперссылок, задаваемый параметром `colorlinks`, не удастся, зато корректировать их цвета можно в любом месте.

В заключение добавим, что `hyperref` позволяет заполнить pdf-форму с информацией о документе, содержащую следующие поля: `pdftitle`, `pdfauthor`, `pdfsubject`, `pdfkeywords`, `pdfcreator`, `pdfproducer`. В

последние два поля автоматически заносятся название компилятора и информация об использованном программном обеспечении. Например, для данной книги это `pdfcreator = {pdfTeX-1.40.21}` и `pdfproducer = {LaTeX with hyperref}`. Остальные поля авторы могут заполнить сами, используя команду `\hypersetup`:

```
\hypersetup{
  pdftitle = {Основы LaTeX},
  pdfauthor = {А.В.Кузнецов},
  pdfsubject = {учебное пособие},
  pdfkeywords = {LaTeX} }
```

12.7. Пакеты коллекции $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$

Коллекцию пакетов $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ составляют пакеты `amsmath`, `amssymb`, `amsbsy`, `amscd`, `amsonp`, `amstext`, `amsthm`, `amsfonts`, `eucal`. Для доступа к основным ресурсам коллекции, обсуждавшимся в гл. 4, достаточно загрузить пакеты `amsmath` и `amssymb`, так как первый автоматически загружает `amsbsy`, `amsonp`, `amstext`, а последний — `amsfonts`.

Перечислим кратко назначение пакетов. Пакеты `amssymb`, `amsfonts` и `eucal` вводят большое количество символов, а также шрифты `\mathbb`, `\mathfrak` и `\mathscr`. Пакеты `amscd` и `amsthm` обеспечивают верстку коммутативных диаграмм и теорем. Пакет `amstext` определяет команду `\text`. Пакет `amsonp` добавляет ряд функций и команды `\operatorname`, `\DeclareMathOperator`. Пакет `amsbsy` вводит команду `\boldsymbol` для печати жирных символов, а также команду

$$\backslash\mathrm{\{выражение\}},$$

которой можно воспользоваться, если они в шрифтах отсутствуют. Данная команда генерирует их путем наложения друг на друга обычных символов, печатаемых с небольшим смещением.

Пакет `amsmath` определяет большое число математических конструкций, а также окружения для создания блоков и нумеро-

ванных формул. Он имеет настройки, управляющие оформлением некоторых элементов формул.

`centertags` | `tbtags` — размещение номера уравнения, содержащего блок `split`. Параметр `centertags` центрирует номер по высоте относительно строк уравнения. При использовании `tbtags` номер проставляется в последней строке, если он выводится справа, или в первой строке, если уравнение нумеруется слева. `split`

`intlimits` | `nointlimits` — размещение пределов интегралов в вынесенных формулах. Параметр `intlimits` расставляет пределы сверху и снизу, а `nointlimits` — сбоку от знака интеграла.

`sumlimits` | `nosumlimits` — аналогичное регулирование пределов остальных больших операторов (см. разд. 4.1.3).

`namelimits` | `nonamelimits` — аналогичное размещение пределов функций `\det`, `\gcd`, `\inf`, `\max`, `\min`, `\Pr` и пределов `\lim`, `\injlim`, `\liminf`, `\limsup` и `\projlim`.

`leqno` | `reqno` — вывод номера слева или справа от формулы.

`fleqn` — выравнивание формул по левому краю страницы.

`alignedleftspaceyes` | `alignedleftspace` | `alignedleftspaceyesifneg` — управление тонким пробелом `\`, перед блоками `aligned` и `gathered`. Первые два параметра, соответственно, вставляют и *не* вставляют его, а третий вставляет `\`, лишь в том случае, если перед блоками находится отрицательный пробел.

Параметры `leqno` и `fleqn` являются аналогами параметров стандартных классов (см. с. 24).

По умолчанию используется набор параметров, реализующий параметры верстки, описанные в гл. 4, а именно: `centertags`, `nointlimits`, `sumlimits`, `namelimits`, `alignedleftspaceyesifneg`.

Заключение

Концепция программной верстки начала развиваться, когда графические возможности мониторов были крайне ограничены. Рост производительности компьютеров и появление дисплеев с высоким разрешением породили концепцию визуальной верстки WYSIWYG — What You See Is What You Get, лежащую в основе офисных редакторов типа *MicrosoftWord*. Каждая концепция имеет свои преимущества и недостатки.

Преимуществом визуальной верстки является ее наглядность, значительно облегчающая редактирование. Она достигается тесной сильной связью документа с конфигурацией программного обеспечения, установленного на компьютере автора, поэтому воспроизведение на другом компьютере или копирование из одного документа в другой часто вызывает трудности: изменяются шрифты, рисунки «уплывают» из отведенного им места, искажаются ячейки в таблицах, формулы зачастую становятся не редактируемыми. Формат документов не стандартизован, и при изменении стиля верстки большую часть работы придется делать заново. Перечисленные недостатки не мешают редактированию офисной документации и простых текстов, поэтому WYSIWYG редакторы стали привычным инструментом, широко представленным на рынке программного обеспечения. Коммерциализация препятствует стандартизации визуальной верстки, поскольку в целях извлечения прибыли в обновления офисных программ часто закладывается какая-либо несовместимость с предыдущими версиями.

Преимуществами программной верстки средствами *L^AT_EX* являются воспроизводимость документа практически на любом компьютере, высокая совместимость частей разных документов, гибкость настроек параметров верстки и очень простое изменение ее стиля, практически не требующее переделки самого документа. Добавим к этому высокое качество верстки формул и удобство работы с перекрестными ссылками и библиографией.

Недостатком является слабая поддержка формата `unicode` в нынешнем стандарте $\LaTeX 2_\epsilon$, вынуждающая представлять символы командами, сильно загромождающими формулы. Эту проблему решает использование современных компиляторов $X\TeX$ и \LuaTeX , разбирающих коды `unicode`. Разработанные для них форматы $X\TeX\LaTeX$ и $\LuaTeX\LaTeX$ вместе с пакетами `fontspec` [51] и `unicode-math` [52] позволяют избавить программу от команд-символов, однако здоровый консерватизм пока сдерживает издателей от введения их в свою практику.



Отмеченный недостаток устраняет графическая оболочка LyX , основанная на концепции WYSIWYM — What You See Is What You Mean, адаптирующей технологии визуальной верстки для \LaTeX . Свободно распространяемая программа LyX , реализована для всех операционных систем и активно развивается. Она использует собственный формат документа, позволяющий заменить команды \LaTeX их графическим представлением, т. е. вместо команды `\section` в окне редактора отражается название раздела в соответствующем шрифтовом оформлении, а формулы имеют графический вид и редактируются с помощью различных меню. Логика программной верстки сохраняется, но набор программы существенно облегчается и ускоряется.

Программу \LaTeX можно импортировать в LyX , а по окончании редактирования экспортировать обратно. Так как синтаксис \LaTeX очень гибок, код, конечно, не будет оптимальным, более того, он может содержать нестандартные решения, поскольку экспорт основан не на оптимальных, а на универсальных алгоритмах, но это — плата за квазивизуальную верстку.

Заинтересованный читатель найдет LyX на сайте www.lyx.org или в пакетах своей операционной системы. Обширная документация LyX имеет собственную wiki-страницу по адресу wiki.lyx.org. Следует, однако, помнить, что конечная цель автора — генерация кода \LaTeX , пригодного для представления в издательство, поэтому для осознанной и успешной работе в LyX нужно сначала освоить \LaTeX , хотя бы в объеме первой части этой книги.

Приложение А

TeXническая информация

Графическая оболочка служит для набора рукописи и организует цепочку ее компиляции, запуская другие программы. В программе **TeXstudio** средства компиляции находятся в меню «Tools». Его строка «Build & View» и иконка  панели инструментов (см. рис 1.1) запускают полную сборку документа, включающую создание оглавления, библиографии и предметного указателя, в которой участвуют несколько компиляторов. По окончании компиляции открывается окно просмотра pdf-документов или программа, показывающая готовый документ, если он имеет другой формат. Строка «Compile» и иконка  запускают используемый по умолчанию компилятор L^AT_EX и, при необходимости, постпроцессор. Строки «Bibliography», «Glossary» и «Index» запускают программы, формирующие библиографию, глоссарий и предметный указатель. Различные варианты компиляции документа и его компонент настраиваются в меню «Options ⇒ Configure TeXstudio... ⇒ Build», а настройки компиляторов содержит меню «Options ⇒ Configure TeXstudio... ⇒ Commands». В меню «Tools ⇒ Commands» можно по отдельности запустить каждую из программ.

Если компиляция выявила ошибки, TeXstudio открывает окно для их анализа и устранения. В сложных случаях, когда оболочка не справляется с обработкой результатов компиляции, ее проще провести вручную. Рассмотрим, как это делается.

Программы, участвующие в создании документа, графического интерфейса не имеют. Их запускают в консоли из командной строки. В Unix-подобных системах открыть консоль не проблема, однако многие пользователи ОС Windows даже не подозревают о ее существовании, поэтому дадим краткую инструкцию.

Компиляцию нужно проводить в рабочей папке, в которой находится документ. Откройте проводник и перейдите в рабочую папку. После этого в окне проводника нажмите «Shift» и правую кнопку мыши. В появившемся меню строку выберите «Открыть окно команд», или что-то подобное.¹ В папке, в которой вы находитесь, откроется консоль. Чтобы настроить ее нажмите на строку заголовка окна консоли правой клавишей мыши и выберите строку «Свойства» в выпавшем меню. Установите комфортный для глаз шрифт и ширину окна не менее 100-120 символов, чтобы в его строке полностью умещались сообщения запускаемых программ. Нажав клавишу «OK» или «Сохранить», можно приступить к работе.

Опишем наиболее сложный процесс интерактивной работы компилятора L^AT_EX, взяв для примера программу pdf_latex.

Чтобы выполнить компиляцию, наберите имя программы pdf_latex (ее расширение .exe в Windows можно опустить) и введите имя компилируемого документа. Начав набор имени, попробуйте нажать клавишу «Tab» клавиатуры, инициирующую автоматическое завершение ввода имени. Программа консоли (оболочка) просматривает имена файлов в рабочей папке и подставляет их в командную строку. Если несколько файлов имеют сходные имена, на консоль выводится их общая часть, а при повторном нажатии «Tab» — подсказка в виде списка имен. Закончив ввод имени, нажмите клавишу «Enter» и запустите компиляцию.

¹ Название строки зависит от версии ОС Windows.

В процессе компиляции `pdflatex` будет печатать в консоли сообщения обо всех загружаемых пакетах, шрифтах и других файлах, а обнаружив ошибку, остановится и напечатает сообщение, начинающейся восклицательным знаком. Оно содержит описание ошибки, за которым следует номер и часть строки компилируемого файла, содержащую ошибку. Перед номером печатается буква «l» и точка, например:

```
! Undefined control sequence.  
1.26 ...и обнаружив ошибку, \ErrorSample  
?
```

В данном сообщении говорится, что команда `\ErrorSample` не определена. Полный список сообщений об ошибках и советы по их устранению приведены в книге [3]. Введя букву «h» и нажав «Enter», можно получить более подробный комментарий к ошибке. При вводе «?» и «Enter» печатается справка о возможных дальнейших действиях.

Изучив ошибку, лучше сразу ее исправить, так как она может служить причиной следующих, после чего нужно нажать клавишу «Enter», чтобы продолжить компиляцию. Если ошибок много, можно ввести букву «s» и «Enter», тогда компиляция продолжится без остановок и закончится генерацией файла с документом. Однако если ошибок слишком много, или одна из них не поддается обработке, компиляция может прерваться, при этом программа не завершит работу и будет ждать дальнейших указаний. Тогда ее нужно остановить принудительно, введя букву «x» и «Enter». В случае аварийной остановки файл с документом не генерируется.

После окончания компиляции все сообщения программы сохраняются в текстовый файл с расширением `•.log`, который можно просмотреть в программе `Notepad` или подобной ей.

Программы, формирующие библиографию и указатели, интерактивную компиляцию не поддерживают. Свою работу они также комментируют сообщениями, посылаемыми на консоль и сохраняемыми в файлах.

Вручную компиляцию лучше запускать без ключей. При необходимости их можно добавить в командную строку, перечислив через пробел без запятых между именем программы и именем файла. Программы дистрибутивов $\text{T}_{\text{E}}\text{X}$, запускаемые из командной строки с ключом `--help`, печатают список ключей, кратко комментируя их назначение, например:

```
bibtex --help
```

```
Usage: bibtex [OPTION]... AUXFILE[.aux]
```

```
Write bibliography for entries in AUXFILE to AUXFILE.bbl,
along with a log file AUXFILE.blg.
```

```
-min-crossrefs=NUMBER include item after NUMBER cross-refs;
                        default 2
-terse                do not print progress reports
-help                 display this help and exit
-version              output version information and exit
```

Список ключей, используемых при запуске программ графической оболочкой TeXstudio , можно посмотреть и настроить в меню «Options \Rightarrow Configure TeXstudio \Rightarrow Commands».

Для обработки документа компиляторы загружают большое количество файлов. Чтобы ускорить их поиск создаются базы данных, которые обслуживает программа `mktexlsr`. При запуске с ключом `--verbose` она выводит на консоль список путей к файлам с базами, например так выглядят ее сообщения в дистрибутиве MiKTeX :

```
mktexlsr --verbose
```

```
Creating fndb for user root directory
```

```
(C:\Users\user\AppData\Roaming\MikTeX\2.9)...
```

```
Creating fndb for user root directory
```

```
(C:\Users\user\AppData\Local\MikTeX\2.9)...
```

```
Skipping common root directory
```

```
(C:\ProgramData\MikTeX\2.9)...
```

```
Skipping common root directory
```

```
("C:\Program Files\MikTeX\2.9")...
```

Первых два сообщают об обновлении баз в папках пользователя. В двух последних написано, что системные папки пропущены.

В папке `...\Roaming\...` находятся пакеты, автоматически устанавливаемые, когда пользователь компилирует документ, или те, что он добавляет сам. Папка `...\Local\...` содержит конфигурации пакетов и программ, сделанные пользователем или от его имени. При обновлении дистрибутива \TeX файлы, находящиеся в этих папках, сохраняются.

В папке `"...\Program Files\..."` находятся пакеты, устанавливаемые администратором. Они доступны для работы всем пользователям, которые не могут изменить что-либо в этой папке.

Обратите внимание на кавычки, охватывающие путь к файлам. Пробелы содержащиеся в *полном* имени файла, включающем путь к нему, должны быть защищены. Единичный пробел можно защитить символом «`\`», например `Program\ Files`. Альтернативной служит совокупная защита всех пробелов двойными кавычками, в которые заключается имя, или его часть.²

Папка `...\ProgramData\...` содержит системные конфигурации пакетов и программ.

Посмотрим теперь структуру папок дистрибутива `TeXLive`:

```
mktxlsr --verbose
mktxlsr: /usr/local/texlive/2020/texmf-config:
    directory not writable. Skipping...
mktxlsr: /usr/local/texlive/2020/texmf-dist:
    directory not writable. Skipping...
mktxlsr: /usr/local/texlive/2020/texmf-var:
    directory not writable. Skipping...
mktxlsr: Updating /home/user/.texlive2020/texmf-var/ls-R...
mktxlsr: Updating /home/user/.texlive2020/texmf-local/ls-R...
mktxlsr: Done.
```

Системные файлы находятся в папке `/usr/local/texlive/2020`, а пользовательские — в папке `/home/user/.texlive2020`. В соот-

² Если возникают проблемы с компиляцией документа, проверьте, не содержит ли пробелы его *полное* имя.

ветствии с назначением, папки разбиты на 3 категории: `config` — настройки, `dist` — пакеты и `var` — файлы, созданные программой, для обеспечения своей работы, например пискельные шрифты. Папка пользователя `.../texmf-local` содержит и пакеты, и настройки.

Папки `.../texmf-●`, название которых происходит от программ `TeX` и `metafont`, созданных Д. Кнутом, тоже организованы в соответствии с назначением файлов. Находящиеся в них папки названы именами программ и содержат файлы, обслуживающие их работу, например `.../texmf-●/tex` или `.../texmf-●/xindy`. Папка `../tex` в свою очередь содержит папку `.../tex/latex`, в которой находятся папки с пакетами. По аналогичному принципу построены и папки `.../MikTeX/2.9` дистрибутива `MikTeX`.

В зависимости от параметров установки дистрибутивов `TeX` и версии операционной системы адреса системных и пользовательских папок изменяются. Чтобы уточнить их, выполните команду `mktexlsr --verbose` на своем компьютере.

Поиск программных средств, обеспечивающих компиляцию, начинается с рабочей папки, затем следуют папки пользователя, обслуживаемые программой `mktexlsr`, в последнюю очередь просматриваются системные папки. Если в компиляции используются средства, отсутствующие в дистрибутиве `TeX`, их можно поместить в одну из папок пользователя и обновить базы, запустив программу `mktexlsr`.

Программа `kpsewhich` позволяет проверить, установлен ли тот или иной файл. Она показывает полное имя файла вместе с путем к нему, например:

```
kpsewhich article.sty
C:/Program Files/MikTeX/2.9/tex/latex/base/article.sty
```

В дистрибутиве `MikTeX` пакеты устанавливаются вместе с документацией, находящейся в системной или пользовательской папках `doc`, имеющих ту же структуру, что папки `texmf-●`. Дистрибутив `TeXLive` документацию не устанавливает, но его реализации в различных Linux-системах, как правило, это делают.

Самые свежие версии пакетов содержит сайт
www.ctan.org,
на котором организован их поиск. Сами пакеты и их документация
находятся на этом же сайте по адресам
www.ctan.org/pkg/имя_пакета.

Приложение Б

ТЕКСТОВЫЕ СИМВОЛЫ

Б.1. Кодировки шрифтов

Универсальной кодировкой текста является `unicode`. Она содержит буквы всех языков вместе с множеством других символов и при 32-разрядной адресации может вместить 4294967296 символов. Вследствие малой разрядности процессоров раньше использовались различные 8-битные кодировки, вмещавшие всего 256 символов, из которых первые 128 являются общими для всех кодировок — ASCII символы, а еще 128 предназначены для букв национальных алфавитов. Среди наиболее часто использовавшихся кодировок выделим `cp1252` (Windows) и `iso8859-1` (международная), а также `cp1251` (Windows). Две первые включают буквы всех европейских языков на основе латиницы, а последняя, обеспечивающая поддержку русского языка, до сих пор используется в русифицированных версиях Windows.

Набор доступных при компиляции символов определяют кодировка программы и набор используемых шрифтов. Чтобы введенный символ попал в печатный документ, компилятор должен определить его привязку к шрифтам. Каждый шрифт может

содержать сотни и тысячи символов, но компилятор `pdflatex` способен работать только с 256-символьными кодировками. В то же время он может оперировать множеством кодировок, поэтому реальный шрифт разбивается на несколько наборов-кодировок, и чтобы охватить все символы шрифта, нужно загрузить несколько кодировок.

Шрифты `Computer Modern` и кодировка `utf8` содержат буквы алфавитов всех европейских языков и большое число небуквенных символов. Кодировку текста `utf8` перекрывает следующий набор кодировок шрифтов: `TS1` — небуквенные символы; `T1` — все буквы латиницы; `T2A`, `T2B`, `T2C`, `X2` и `OT2` — различные наборы букв кириллицы. Символы кодировок `T1` и `T2A`, загружаемых по умолчанию, приведены в табл. Б.1

Кодировки шрифтов загружает пакет:

```
\usepackage[список кодировок]{fontenc} .
```

Кодировки перечисляются списком через запятую в параметре загружающей его команды. Последняя из них служит основной кодировкой, символы которой используются по умолчанию. Символы остальных кодировок нужно дополнительно объявлять с помощью команд

```
\DeclareTextSymbolDefault{команда}{кодировка} .
```

Объявленные символы вводятся в текст обычным образом посредством кодов или команд. При объявлении основной буквы можно использовать все ее акцентированные варианты.

Если символ используется редко, команда

```
\UseTextSymbol{кодировка}{команда} .
```

позволяет вставить его без объявления.

Поясним сказанное примером. Чтобы составить полную таблицу кириллицы, использовалась загрузка:

```
\usepackage[OT2,X2,T2B,T2C,T2A]{fontenc},
```

а затем в преамбуле объявлялись буквы, отсутствующие в кодировке `T2A`. Например, фита была объявлена командой

```
\DeclareTextSymbolDefault{\CYRFITA}{OT2} .
```


Таблица кодировок T1 и T2A¹

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0●	‘	’	^	”	”	°	˘	˘	—	·	˘	˘	˘	˘	˘	˘
1●	“	”	”	»	»	—	—	—	°	ı	j	ff	fi	fl	ffi	ffl
2●	˘	!	”	#	\$	%	&	’	()	*	+	,	-	.	/
3●	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4●	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5●	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6●	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7●	p	q	r	s	t	u	v	w	x	y	z	{		}	~	-
8●	Ǻ	Ǻ	Č	Č	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ
9●	Ř	Š	Š	Š	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ
A●	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ
B●	ř	š	š	š	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ	Ǻ
C●	À	Á	Â	À	Ä	À	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D●	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	Ø	Ù	Ú	Û	Ü	Ý	Þ	ŠŠ
E●	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F●	đ	ñ	ò	ó	ô	õ	ö	œ	ø	ù	ú	û	ü	ý	þ	ß
8●	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г
9●	Ө	Ç	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ
A●	г	г	г	г	г	г	г	г	г	г	г	г	г	г	г	г
B●	ө	ç	ÿ	ÿ	ÿ	ÿ	ÿ	ÿ	ÿ	ÿ	ÿ	ÿ	ÿ	ÿ	ÿ	ÿ
C●	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D●	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E●	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F●	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

¹ Символы с кодами 80–FF в средней и нижней части таблицы содержат, соответственно, кодировки T1 и T2A. Символы с кодами 00–7F, кроме подчеркнутых, являются общими. Среди подчеркнутых левые входят в кодировку T1, а правые — T2A

Разберем теперь, как формируются ресурсы, обеспечиваемые минимальной загрузкой пакетов, представленной на с. 26. Загружаемая пакетом `inputenc` кодировка текста `utf8` открывает доступ к 522 символам, которые можно использовать при наборе в дополнение к 94 символам стандартной клавиатуры. Ядро `LaTeX` автоматически загружает кодировку шрифта `T1`, обеспечивая поддержку букв латиницы. Для русского языка пакет `babel` автоматически загружает кодировку `T2A`. Пакет `textcomp` загружает кодировку `TS1` и привязывает к ней небуквенные символы. И наконец, чтобы обеспечить поиск и копирование текста в сверстанном документе, пакет `cmap` подгружает в него таблицу соответствия символов кодам `unicode`. Его следует загружать непосредственно после загрузки кодировки текста программы.

Здесь рассмотрены лишь стандартные ресурсы, обеспечиваемые кодировкой `utf8` и шрифтами `Computer Modern`. Значительное число нестандартных пакетов позволяет расширить набор символов путем загрузки дополнительных шрифтов. Часть из них описана в книгах [3, 4].

Б.2. Текстовые символы

В колонках табл. Б.2, маркированных значком \bullet , представлены текстовые символы стандартных гарнитур `Computer Modern`, доступные в кодировке `utf8` после загрузки пакета `textcomp`. В колонках с заголовком `U+#` приведены шестнадцатеричные коды символов, а следующих колонках — соответствующие им команды.

Текстовые символы Таблица Б.2

\bullet	U+#	Команды	\bullet	U+#	Команды
	200c	<code>\textcompwordmark</code>		a0	<code>\nobreakspace</code>
/	2044	<code>\textfractionsolidus</code>	\		<code>\textbackslash</code>
	2423	<code>\textvisiblespace</code>	°	b0	<code>\textdegree</code>

•	U+#	Команды	•	U+#	Команды
<	2039	<code>\guilsinglleft</code>	μ	b5	<code>\textmu</code>
>	203a	<code>\guilsinglright</code>	×	d7	<code>\texttimes</code>
«	ab	<code>\guillemotleft</code>	÷	f7	<code>\textdiv</code>
»	bb	<code>\guillemotright</code>	±	b1	<code>\textpm</code>
*	204e	<code>\textasteriskcentered</code>	№	2116	<code>\textnumero</code>
·	b7	<code>\textperiodcentered</code>	...	2026	<code>\textellipsis</code>
○	25e6	<code>\textopenbullet</code>	●	2022	<code>\textbullet</code>
♪	266a	<code>\textmusicalnote</code>	○	25ef	<code>\textbigcircle</code>
	a6	<code>\textbrokenbar</code>		2016	<code>\textbardbl</code>
←	2190	<code>\textleftarrow</code>	<	2329	<code>\textlangle</code>
→	2192	<code>\textrightarrow</code>	>	232a	<code>\textrangle</code>
↑	2191	<code>\textuparrow</code>	†	2020	<code>\textdagger</code>
↓	2193	<code>\textdownarrow</code>	‡	2020	<code>\dag</code>
¿	bf	<code>\textquestiondown</code>	‡	2021	<code>\textdaggerdbl</code>
¡	a1	<code>\textexclamdown</code>	‡	2021	<code>\ddag</code>
?	203d	<code>\textinterrobang</code>	?	e3f	<code>\textbaht</code>
,	201a	<code>\quotesinglbase</code>	„	201e	<code>\quotedblbase</code>
‘	2018	<code>\textquoteleft</code>	ℵ	20a6	<code>\textnaira</code>
’	2019	<code>\textquoteright</code>	₪	20b1	<code>\textpeso</code>
“	201c	<code>\textquotedblleft</code>	₩	20a9	<code>\textwon</code>
”	201d	<code>\textquotedblright</code>	₫	20ab	<code>\textdong</code>
‰	2030	<code>\textperthousand</code>	€	20ac	<code>\texteuro</code>
‱	2031	<code>\textpertenthousand</code>	¥	a5	<code>\textyen</code>
%	2052	<code>\textdiscount</code>	¢	a2	<code>\textcent</code>
⌠	20a1	<code>\textcolonmonetary</code>	£	20a4	<code>\textlira</code>
¤	a4	<code>\textcurrency</code>	£	a3	<code>\textsterling</code>
ª	aa	<code>\textordfeminine</code>	f	192	<code>\textflorin</code>
º	ba	<code>\textordmasculine</code>	¬	ac	<code>\textlnot</code>
¹	b9	<code>\textonesuperior</code>	Ω	2126	<code>\textohm</code>
²	b2	<code>\texttwosuperior</code>	ℳ	2127	<code>\textmho</code>
³	b3	<code>\textthreesuperior</code>	€	212e	<code>\textestimated</code>

•	U+#	Команды	•	U+#	Команды
$\frac{1}{2}$	bd	<code>\textonehalf</code>	°C	2103	<code>\textcelsius</code>
$\frac{1}{4}$	bc	<code>\textonequarter</code>	¶	b6	<code>\textparagraph</code>
$\frac{3}{4}$	be	<code>\textthreequarters</code>	§	a7	<code>\textsection</code>
®	ae	<code>\textregistered</code>	ℬ	2422	<code>\textblank</code>
©	a9	<code>\textcopyright</code>	℞	211e	<code>\textrecipe</code>
Ⓟ	2117	<code>\textcircledP</code>	⊙		<code>\textcircled{•}</code>
SM	2120	<code>\textservicemark</code>	™	2122	<code>\texttrademark</code>
※	203b	<code>\textreferencemark</code>	-	2010	-
—	2014	<code>\textemdash, ---</code>	—	2013	<code>\textendash, --</code>
¨	a8	<code>\textasciidieresis</code>	”	2dd	<code>\textacutedbl</code>
—	af	<code>\textasciimacron</code>	~	2dc	<code>\textasciitilde</code>
’	b4	<code>\textasciiacute</code>	˘	2d8	<code>\textasciibreve</code>
^	2c6	<code>\textasciicircum</code>	ˇ	2c7	<code>\textasciicaron</code>

Б.3. Латиница

Большое количество букв латиницы и кириллицы имеют диакритические знаки, расставляемые с помощью команд, собранных в табл. Б.3. В колонках, помеченных символом •, приведены примеры акцентированных букв.

Таблица Б.3

Команды расстановки диакритических знаков

•	Команды	•	Команды	•	Команды	•	Команды
à	<code>\‘a</code>	á	<code>\’a</code>	ä	<code>\"a</code>	â	<code>\^a</code>
ā	<code>\=a</code>	à	<code>\.a</code>	ã	<code>\C{a}</code>	â	<code>\f{a}</code>
â	<code>\r{a}</code>	ă	<code>\U{a}</code>	ǎ	<code>\u{a}</code>	ǎ	<code>\v{a}</code>
ã	<code>\~a</code>	ǎ	<code>\H{a}</code>	ââ	<code>\t{aa}</code>		
â	<code>\b{a}</code>	ȁ	<code>\c{a}</code>	ȁ	<code>\d{a}</code>	ȁ	<code>\k{a}</code>
ȁ	<code>\textcommabelow{a}</code>						

Буквы латиницы представлены в табл. Б.4.

Таблица Б.4

Буквы алфавитов латиницы

•	U+#	Команды	•	U+#	Команды	•	U+#	Команды	•	U+#	Команды
À	C0	\‘A	à	E0	\‘a	Á	C1	\’A	á	E1	\’A
Â	C2	\^A	â	E2	\^a	Ã	C3	\~A	ã	E3	\~a
Ä	C4	\"A	ä	E4	\"A	Å	C5	\r{A}	å	E5	\r{a}
Ā	100	\=A	ā	101	\=a	Ă	102	\u{A}	ă	103	\u{a}
Ǻ	1CD	\v{A}	ǻ	1CE	\u{a}	Ą	104	\k{A}	ą	105	\k{a}
Æ	C6	\AE	æ	E6	\ae	Ǽ	1E2	\=\AE	ǽ	1E3	\=\ae
Ĭ	1E02	\.B	ĭ	1E03	\.b	Ć	106	\’C	ć	107	\’c
Ĉ	108	\^C	ĉ	109	\^c	Ċ	10A	\.C	ċ	10B	\.c
Č	10C	\v{C}	č	10D	\v{c}	Ç	C7	\c{C}	ç	E7	\c{c}
Ǫ	10E	\v{D}	ǫ	10F	\v{d}	Ð	D0	\DH	ð	F0	\dh
Đ	110	\DJ	đ	111	\dj	È	C8	\‘E	è	E8	\‘e
É	C9	\’E	é	E9	\’e	Ê	CA	\^E	ê	EA	\^e
Ë	CB	\"E	ë	EB	\"e	Ē	112	\=E	ē	113	\=e
Ĕ	116	\.E	ĕ	117	\.e	Ë	114	\u{E}	ë	115	\u{e}
Ě	11A	\v{E}	ě	11B	\v{e}	Ě	118	\k{E}	ě	119	\k{e}
Ĝ	1F4	\’G	ĝ	1F5	\’g	Ġ	120	\.G	ġ	121	\.g
Ĝ	1E20	\=E	ġ	1E21	\=g	Ģ	11C	\^G	ģ	11D	\^g
Ģ	11E	\u{G}	ģ	11F	\u{g}	Ģ	1E6	\v{G}	ģ	1E7	\v{g}
Ģ	122	\c{G}	g	123	\c{g}	Ĥ	124	\^H	ĥ	125	\^h

•	U+#	Команды	•	U+#	Команды	•	U+#	Команды	•	U+#	Команды
Ì	CC	\‘I	ì	EC	\‘i	Í	CD	\’I	í	ED	\’i
Î	CE	\^I	î	EE	\^i	Ï	CF	\"I	ï	EF	\"i
Ĩ	128	\~I	ĩ	129	\~i	Ī	12A	\=I	ī	12B	\=i
İ	130	\.I	ı	131	\i	Ĭ	12C	\u{I}	ÿ	12D	\u{i}
Ĳ	1CF	\v{I}	ÿ	1D0	\v{i}	Ĵ	12E	\k{I}	ĵ	12F	\k{i}
IJ	132	\IJ	ij	133	\ij	Ĵ	134	\^J	ĵ	135	\^j
J	237	\j	ĵ	1F0	\v{j}	Ķ	1E8	\v{K}	ķ	1E9	\v{k}
Ɔ	136	\c{K}	ķ	137	\c{k}	Ĺ	139	\’L	ĺ	13A	\’l
Ł	13D	\v{L}	ł	13E	\v{l}	Ľ	13B	\c{L}	ľ	13C	\c{l}
Ł	141	\L	ł	142	\l	Ń	143	\’N	ń	144	\’n
Ń	D1	\~N	ń	F1	\~n	Ń	147	\v{N}	ň	148	\v{n}
Ń	145	\c{N}	ņ	146	\c{n}	Ŋ	14A	\NG	ŋ	14B	\ng
Ŏ	D2	\‘O	ò	F2	\‘o	Ó	D3	\’O	ó	F3	\’o
Ô	D4	\^O	ô	F4	\^i	Õ	D5	\~O	õ	F5	\~o
Ö	D6	\"O	ö	F6	\"o	Ȫ	14C	\=O	ȫ	14D	\=o
Ȫ	14E	\u{O}	ȫ	14F	\u{o}	Ȭ	150	\H{O}	ȭ	151	\H{o}
Ȭ	1D1	\v{O}	ȭ	1D2	\v{o}	Ȯ	1EA	\k{O}	ȯ	1EB	\k{o}
Ø	D8	\O	ø	F8	\o	Œ	152	\OE	œ	153	\oe
Ŕ	154	\’R	ŕ	155	\’r	Ř	158	\v{R}	ř	159	\v{r}
Ŗ	156	\c{R}	ŗ	157	\c{r}	Ś	15A	\’S	ś	15B	\’s

•	U+#	Команды	•	U+#	Команды	•	U+#	Команды	•	U+#	Команды
Ŝ	15C	\^S	ŝ	15D	\^s	Š	160	\v{S}	š	161	\v{s}
Ş	15E	\c{S}	ş	15F	\c{s}				ß	DF	\ss
Ș	218	\textcommabelow{S}				ș	219	\textcommabelow{s}			
Ť	21A	\textcommabelow{T}				ť	21B	\textcommabelow{t}			
Ť	164	\v{T}	ť	165	\v{t}	Ť	162	\c{T}	ť	163	\c{t}
Þ	DE	\TH	þ	FE	\th	Û	D9	\^U	ù	F9	\^u
Ú	DA	\'U	ú	FA	\'u	Û	DB	\^U	û	FB	\^u
Û	DC	\"U	ü	FC	\"u	Ũ	168	\~U	ũ	169	\~u
Ū	16A	\=U	ū	16B	\=u	Ũ	16C	\u{U}	ũ	16D	\u{u}
Ů	16E	\r{U}	ů	16F	\r{u}	Ů	170	\H{U}	ů	171	\H{u}
Ů	1D3	\v{U}	ů	1D4	\v{u}	Ů	172	\k{U}	ů	173	\k{u}
Ŵ	174	\^W	ŵ	175	\^w	Ŷ	DD	\'Y	ŷ	FD	\'y
Ŷ	176	\^Y	ŷ	177	\^y	ÿ	178	\"Y	ÿ	FF	\"y
Ȳ	232	\=Y	ȳ	233	\=y	Ž	179	\'Z	ž	17A	\'z
Ẑ	17B	\.Z	ẑ	17C	\.z	Ž	17D	\v{Z}	ž	17E	\v{z}

Б.4. Кириллица

В табл. Б.5 собраны буквы кириллицы. Их поддержку обеспечивает несколько кодировок шрифтов, указанных в последней колонке с помощью следующих сокращений: А — Т2А, В — Т2В, С — Т2С, О — ОТ2, Х — Х2. Таким образом аббревиатура АВОХ означает, что буквы содержатся в кодировках Т2А, Т2В, ОТ2 и Х2. Буквам, помещенным в колонки, маркированные символом ●, соответствуют команды `\IeC{*** }`, где звездочки обозначают команды, указанные в таблице. В колонках с заголовком U+# представлены коды символов в кодировке unicode.

Буквы алфавитов кириллицы

Таблица Б.5

●	U+#	Команды	●	U+#	Команды	Кодировки
А	410	<code>\CYRA</code>	а	430	<code>\cyra</code>	АВСОХ
Ӑ	4D0	<code>\U\CYRA</code>	ӓ	4D1	<code>\U\cyra</code>	АВСОХ
Ӓ	4D2	<code>\"\CYRA</code>	Ӕ	4D3	<code>\"\cyra</code>	АВСОХ
Ӕ	4D4	<code>\CYRAE</code>	ӕ	4D5	<code>\cyrae</code>	АХ
Б	411	<code>\CYRB</code>	б	431	<code>\cyrb</code>	АВСОХ
В	412	<code>\CYRV</code>	в	432	<code>\cyrv</code>	АВСОХ
Г	413	<code>\CYRG</code>	г	433	<code>\cyrg</code>	АВСОХ
Ґ	403	<code>\'\CYRG</code>	ѓ	453	<code>\'\cyrg</code>	АВСОХ
Ґ	490	<code>\CYRGUP</code>	ѓ	491	<code>\cyrgup</code>	АХ
Ғ	492	<code>\CYRGHCRS</code>	ғ	493	<code>\cyrghcrs</code>	АВХ
Ҙ	494	<code>\CYRGHC</code>	ҙ	495	<code>\cyrghc</code>	ВСХ
Ғ	4F6	<code>\CYRGDSC</code>	ғ	4F7	<code>\cyrgdsc</code>	ВХ
Ғ	4FA	<code>\CYRGDSCCHRS</code>	ғ	4FB	<code>\cyrgdschcrs</code>	В
Д	414	<code>\CYRD</code>	д	434	<code>\cyrd</code>	АВСОХ
Е	415	<code>\CYRE</code>	е	435	<code>\cyre</code>	АВСОХ
È	400	<code>\'\CYRE</code>	è	450	<code>\'\cyre</code>	АВСОХ
Ӑ	4D6	<code>\U\CYRE</code>	ӓ	4D7	<code>\U\cyre</code>	АВСОХ
Ӓ	401	<code>\CYRYO</code>	Ӕ	451	<code>\cyryo</code>	АВСОХ
Ж	416	<code>\CYRZH</code>	ж	436	<code>\cyrzh</code>	АВСОХ

•	U+#	Команды	•	U+#	Команды	Кодировки
Ж̇	4DC	\"\CYRZH	ж̇	4DD	\"\cyrz	ABCOX
Ӝ	4C1	\U\CYRZH	Ӝ	4C2	\U\cyrz	ABCOX
Ж̉	496	\CYRZHSDSC	ж̉	497	\cyrzhdsc	ABX
З̇	417	\CYRZ	з̇	437	\cyrz	ABCOX
Ӟ	4DE	\"\CYRZ	Ӟ	4DF	\"\cyrz	ABCOX
З̉	498	\CYRZDSC	з̉	499	\cyrzdsc	AX
И̇	418	\CYRI	и̇	438	\cyri	ABCOX
Ӥ	40D	\'\CYRI	Ӥ	45D	\'\cyri	ABCOX
И̉	4E2	\=\CYRI	и̉	4E3	\=\cyri	ABCOX
Й̇	4E4	\"\CYRI	Й̇	4E5	\"\cyri	ABCOX
Й̈	419	\CYRISHRT	й̈	439	\cyrishrt	ABCOX
К̇	41A	\CYRK	к̇	43A	\cyrk	ABCOX
К̈	40C	\'\CYRK	К̈	45C	\'\cyrk	ABCOX
К̉	49A	\CYRKDSC	к̉	49B	\cyrkdsc	ABCX
К̊	49C	\CYRKVCRS	к̊	49D	\cyrkvcrs	AX
К̋	49E	\CYRKHCRS	к̋	49F	\cyrkhcrs	AX
К̌	4A0	\CYRKBEAK	к̌	4A1	\cyrkbeak	AX
Қ̇	4C3	\CYRKHK	қ̇	4C4	\cyrkhk	BX
Л̇	41B	\CYRL	л̇	43B	\cyrl	ABCOX
Л̈	4C5	\CYRLDSC	Л̈	4C6	\cyrlsdsc	BCX
Л̉	409	\CYRLJE	л̉	459	\cyrlje	ABOX
М̇	41C	\CYRM	м̇	43C	\cyrm	ABCOX
М̈	4CD	\CYRMDSC	М̈	4CE	\cyrmdsc	CX
Н̇	41D	\CYRN	н̇	43D	\cyrn	ABCOX
Н̈	4A2	\CYRNDSC	Н̈	4A3	\cyrndsc	ABCX
Н̉	4A4	\CYRNG	н̉	4A5	\cyrng	ABX
Н̊	40A	\CYRNJE	Н̊	45A	\cyrnje	ABOX
Н̋	4C7	\CYRNHK	н̋	4C8	\cyrnhk	BCX
О̇	41E	\CYRO	о̇	43E	\cyro	ABCOX
Ö̇	4E6	\"\CYRO	ö̇	4E7	\"\cyro	ABCOX
Ө̇	4E8	\CYROTLD	ө̇	4E9	\cyrotld	ABCX

•	U+#	Команды	•	U+#	Команды	Кодировки
П	41F	\CYRP	п	43F	\cyrp	ABCOX
Ѕ	4A6	\CYRPHK	ѕ	4A7	\cyrphk	CX
Р	420	\CYRR	р	440	\cyr	ABCOX
Р	48E	\CYRRTICK	р	48F	\cyrrtick	C
С	421	\CYRS	с	441	\cyr	ABCOX
Ѕ	4AA	\CYRSDSC	ѕ	4AB	\cyrsdsc	AX
Т	422	\CYRT	т	442	\cyr	ABCOX
Т	4AC	\CYRTDSC	т	4AD	\cyrtdsc	CX
Ц	4B4	\CYRTETSE	ц	4B5	\cyrtetse	CX
У	423	\CYRU	у	443	\cyr	ABCOX
Ū	4EE	\=\CYRU	Ū	4EF	\=\cyr	ABCOX
Û	4F0	\"\CYRU	Û	4F1	\"\cyr	ABCOX
Ÿ	4F2	\H\CYRU	Ÿ	4F3	\H\cyr	ABCOX
Ÿ	40E	\CYRUSRT	Ÿ	45E	\cyrushrt	ABX
Ф	424	\CYRF	ф	444	\cyr	ABCOX
Х	425	\CYRH	х	445	\cyr	ABCOX
Х	4B2	\CYRHDSC	х	4B3	\cyrhdsc	ABCX
Х	4FC	\CYRHHC	х	4FD	\cyrhhc	BX
Х	4FE	\CYRHHCERS	х	4FF	\cyrhhcers	B
Ц	426	\CYRC	ц	446	\cyr	ABCOX
Ч	427	\CYRCH	ч	447	\cyr	ABCOX
Č	4F4	\"\CYRCH	č	4F5	\"\cyr	ABCOX
Ч	4B6	\CYRCHRDSC	ч	4B7	\cyrchrdsc	ABCX
Ч	4CB	\CYRCHLDSC	ч	4CC	\cyrchldsc	BX
Ч	4B8	\CYRCHVCERS	ч	4B9	\cyrchvcers	AX
Ш	428	\CYRSH	ш	448	\cyr	ABCOX
Ш	429	\CYRSHCH	ш	449	\cyrshch	ABCOX
Ъ	42A	\CYRHRDSN	ъ	44A	\cyrhrdsn	ABCOX
Ы	42B	\CYRERY	ы	44B	\cyrery	ABCOX
Û	4F4	\"\CYRERY	Û	4F5	\"\cyrERY	ABCOX
Ь	42C	\CYRSFTSN	ь	44C	\cyr	ABCOX

•	U+#	Команды	•	U+#	Команды	Кодировки
Б	48C	\CYRSEMISFTSN	б	48D	\cyrsemisftsn	C
Э	42D	\CYREREV	э	44D	\cyrerev	ABCOX
Э	4EC	\\"CYREREV	э	4ED	\\"cyrerev	ABCOX
Є	404	\CYRIE	є	454	\cyrie	AOX
Ю	42E	\CYRYU	ю	44E	\cyrju	ABCOX
Я	42F	\CYRYA	я	44F	\cyrja	ABCOX
І	406	\CYRII	і	456	\cyrrii	ABCOX
İ	407	\CYRYI	ï	457	\cyrrii	AX
J	408	\CYRJE	j	458	\cyrje	ABCOX
S	405	\CYRDZE	s	455	\cyrdze	ABCOX
Y	4AE	\CYRY	у	4AF	\cyrju	ABX
Y	4B0	\CYRYHCRS	у	4B1	\cyrjuhcrs	AX
h	4BA	\CYRSHHA	h	4BB	\cyrshha	ABCX
Ђ	402	\CYRDJE	ђ	452	\cyrdje	AOX
Ђ	40B	\CYRTSHE	ђ	45B	\cyrtshe	AOX
Ц	40F	\CYRDZHE	ц	45F	\cyrdze	ACOX
Ə	4D8	\CYRSCHWA	ə	4D9	\cyrschw	ABCX
Ə	4DA	\\"CYRSCHWA	ə	4DB	\\"cyrschw	ABCX
Є	4BC	\CYRABHCH	є	4BD	\cyrabhch	CX
Є	4BE	\CYRABHCHDSC	є	4BF	\cyrabhchdsc	CX
Q	4A8	\CYRABHHA	q	4A9	\cyrabhha	CX
З	4E0	\CYRABHDZE	з	4E1	\cyrabhdze	BCX
Ђ	462	\CYRYAT	ђ	463	\cyrjat	OX
Ж	46A	\CYRBYUS	ж	46B	\cyrbyus	X
Ө	472	\CYRFITA	ө	473	\cyrfita	O
V	474	\CYRIZH	v	475	\cyrizh	OX
Û	474	\C\CYRIZH	Û	475	\C\cyrizh	OX
			I	4C0	\CYRpalochka	ABCX

Приложение В

Математические символы

Математические символы делятся на восемь категорий. Категория определяет размер полей вокруг символа, а также правила его взаимодействия с другими элементами формулы.

Приведенные ниже команды позволяют определить или переопределить категорию символа:

- | | |
|-----------------------------------|------------------------------|
| 0 <code>\mathord{символ}</code> | — обычный символ; |
| 1 <code>\mathop{символ}</code> | — большой оператор, функция; |
| 2 <code>\mathbin{символ}</code> | — бинарная операция; |
| 3 <code>\mathrel{символ}</code> | — соотношение; |
| 4 <code>\mathopen{символ}</code> | — открывающий разделитель; |
| 5 <code>\mathclose{символ}</code> | — закрывающий разделитель; |
| 6 <code>\mathpunct{символ}</code> | — знак пунктуации; |
| 7 <code>\mathalpha{символ}</code> | — буква. |

К категории `\mathalpha` относятся буквы и цифры стандартного математического шрифта, а также прописные греческие буквы. Строчные греческие буквы и буквы математических алфавитов принадлежат категории `\mathord`.

В представленных далее таблицах символы сгруппированы по категориям.

В.1. Математические алфавиты и акценты

В табл. В.1 – В.8 приведены математические алфавиты коллекции шрифтов Computer Modern. Таблицы содержат символы и их шестнадцатеричные коды в кодировке `unicode`. Колонки с символами маркированы значком \bullet , а колонки с кодами — заголовком `U+#`. В заголовках таблиц указаны команды с префиксом `\math`, выводящие символы различных шрифтов.

Команда `\mathit{\bullet}` выводит буквы и цифры математического курсива, представленные в табл. В.1. Для вывода жирных символов используется команда `\boldsymbol{\bullet}`.

Цифры, отсутствующие в кодировке `unicode`, показаны без кодов.

Математический курсив

Таблица В.1

<code>\mathit{\bullet}</code>								
\bullet	U+#	\bullet	U+#	\bullet	U+#	\bullet	U+#	
<i>A</i>	1d434	<i>a</i>	1d44e	<i>N</i>	1d441	<i>n</i>	1d45b	<i>0</i>
<i>B</i>	1d435	<i>b</i>	1d44f	<i>O</i>	1d442	<i>o</i>	1d45c	<i>1</i>
<i>C</i>	1d436	<i>c</i>	1d450	<i>P</i>	1d443	<i>p</i>	1d45d	<i>2</i>
<i>D</i>	1d437	<i>d</i>	1d451	<i>Q</i>	1d444	<i>q</i>	1d45e	<i>3</i>
<i>E</i>	1d438	<i>e</i>	1d452	<i>R</i>	1d445	<i>r</i>	1d45f	<i>4</i>
<i>F</i>	1d439	<i>f</i>	1d453	<i>S</i>	1d446	<i>s</i>	1d460	<i>5</i>
<i>G</i>	1d43a	<i>g</i>	1d454	<i>T</i>	1d447	<i>t</i>	1d461	<i>6</i>
<i>H</i>	1d43b	<i>h</i>	1d455	<i>U</i>	1d448	<i>u</i>	1d462	<i>7</i>
<i>I</i>	1d43c	<i>i</i>	1d456	<i>V</i>	1d449	<i>v</i>	1d463	<i>8</i>
<i>J</i>	1d43d	<i>j</i>	1d457	<i>W</i>	1d44a	<i>w</i>	1d464	<i>9</i>
<i>K</i>	1d43e	<i>k</i>	1d458	<i>X</i>	1d44b	<i>x</i>	1d465	
<i>L</i>	1d43f	<i>l</i>	1d459	<i>Y</i>	1d44c	<i>y</i>	1d466	
<i>M</i>	1d440	<i>m</i>	1d45a	<i>Z</i>	1d44d	<i>z</i>	1d467	

\boldsymbol{\mathit{\{•\}}}									
•	U+#	•	U+#	•	U+#	•	U+#	•	
<i>A</i>	1d468	<i>a</i>	1d482	<i>N</i>	1d475	<i>n</i>	1d48f	<i>0</i>	
<i>B</i>	1d469	<i>b</i>	1d483	<i>O</i>	1d476	<i>o</i>	1d490	<i>1</i>	
<i>C</i>	1d46a	<i>c</i>	1d484	<i>P</i>	1d477	<i>p</i>	1d491	<i>2</i>	
<i>D</i>	1d46b	<i>d</i>	1d485	<i>Q</i>	1d478	<i>q</i>	1d492	<i>3</i>	
<i>E</i>	1d46c	<i>e</i>	1d486	<i>R</i>	1d479	<i>r</i>	1d493	<i>4</i>	
<i>F</i>	1d46d	<i>f</i>	1d487	<i>S</i>	1d47a	<i>s</i>	1d494	<i>5</i>	
<i>G</i>	1d46e	<i>g</i>	1d488	<i>T</i>	1d47b	<i>t</i>	1d495	<i>6</i>	
<i>H</i>	1d46f	<i>h</i>	1d489	<i>U</i>	1d47c	<i>u</i>	1d496	<i>7</i>	
<i>I</i>	1d470	<i>i</i>	1d48a	<i>V</i>	1d47d	<i>v</i>	1d497	<i>8</i>	
<i>J</i>	1d471	<i>j</i>	1d48b	<i>W</i>	1d47e	<i>w</i>	1d998	<i>9</i>	
<i>K</i>	1d472	<i>k</i>	1d48c	<i>X</i>	1d47f	<i>x</i>	1d499		
<i>L</i>	1d473	<i>l</i>	1d48d	<i>Y</i>	1d480	<i>y</i>	1d49a		
<i>M</i>	1d474	<i>m</i>	1d48e	<i>Z</i>	1d481	<i>z</i>	1d49b		

Команда `\mathrm{•}` выводит буквы и цифры математического шрифта прямого начертания, представленные в табл. В.2.

Коды символов нормальной насыщенности соответствуют ASCII-кодировке. Для вывода жирных символов используется команда `\mathbf{•}`.

Прямой шрифт Roman

Таблица В.2

\mathrm{•}									
•	U+#	•	U+#	•	U+#	•	U+#	•	U+#
A	41	a	61	N	4e	n	6e	0	30
B	42	b	62	O	4f	o	6f	1	31
C	43	c	63	P	50	p	70	2	32
D	44	d	64	Q	51	q	71	3	33
E	45	e	65	R	52	r	72	4	34
F	46	f	66	S	53	s	73	5	35

Окончание табл. В.2

•	U+#	•	U+#	•	U+#	•	U+#	•	U+#
G	47	g	67	T	54	t	74	6	36
H	48	h	68	U	55	u	75	7	37
I	49	i	69	V	56	v	76	8	38
J	4a	j	6a	W	57	w	77	9	39
K	4b	k	6b	X	58	x	78		
L	4c	l	6c	Y	59	y	79		
M	4d	m	6d	Z	5a	z	7a		

`\mathbf{•}`

•	U+#	•	U+#	•	U+#	•	U+#	•	U+#
A	1d400	a	1d41a	N	1d40d	n	1d427	0	1d7ce
B	1d401	b	1d41b	O	1d40e	o	1d428	1	1d7cf
C	1d402	c	1d41c	P	1d40f	p	1d429	2	1d7d0
D	1d403	d	1d41d	Q	1d410	q	1d42a	3	1d7d1
E	1d404	e	1d41e	R	1d411	r	1d42b	4	1d7d2
F	1d405	f	1d41f	S	1d412	s	1d42c	5	1d7d3
G	1d406	g	1d420	T	1d413	t	1d42d	6	1d7d4
H	1d407	h	1d421	U	1d414	u	1d42e	7	1d7d5
I	1d408	i	1d422	V	1d415	v	1d42f	8	1d7d6
J	1d409	j	1d423	W	1d416	w	1d430	9	1d7d7
K	1d40a	k	1d424	X	1d417	x	1d431		
L	1d40b	l	1d425	Y	1d418	y	1d432		
M	1d40c	m	1d426	Z	1d419	z	1d433		

Команда `\mathsf{•}` выводит буквы и цифры математического рубленого шрифта, представленные в табл. В.3. Для вывода жирных символов используется команда `\boldsymbol{•}`.

\mathsf{•}

•	U+#	•	U+#	•	U+#	•	U+#	•	U+#
A	1d5a0	a	1d5ba	N	1d5ad	n	1d5c7	0	1d7e2
B	1d5a1	b	1d5bb	O	1d5ae	o	1d5c8	1	1d7e3
C	1d5a2	c	1d5bc	P	1d5af	p	1d5c9	2	1d7e4
D	1d5a3	d	1d5bd	Q	1d5b0	q	1d5ca	3	1d7e5
E	1d5a4	e	1d5be	R	1d5b1	r	1d5cb	4	1d7e6
F	1d5a5	f	1d5bf	S	1d5b2	s	1d5cc	5	1d7e7
G	1d5a6	g	1d5c0	T	1d5b3	t	1d5cd	6	1d7e8
H	1d5a7	h	1d5c1	U	1d5b4	u	1d5ce	7	1d7e9
I	1d5a8	i	1d5c2	V	1d5b5	v	1d5cf	8	1d7ea
J	1d5a9	j	1d5c3	W	1d5b6	w	1d5d0	9	1d7eb
K	1d5aa	k	1d5c4	X	1d5b7	x	1d5d1		
L	1d5ab	l	1d5c5	Y	1d5b8	y	1d5d2		
M	1d5ac	m	1d5c6	Z	1d5b9	z	1d5d3		

\boldsymbol{\mathsf{•}}

•	U+#	•	U+#	•	U+#	•	U+#	•	U+#
A	1d5d4	a	1d5ee	N	1d5e2	n	1d5fb	0	1d7ec
B	1d5d5	b	1d5ef	O	1d5e3	o	1d5fc	1	1d7ed
C	1d5d6	c	1d5f0	P	1d5e4	p	1d5fd	2	1d7ee
D	1d5d7	d	1d5f1	Q	1d5e5	q	1d5fe	3	1d7ef
E	1d5d8	e	1d5f2	R	1d5e6	r	1d5ff	4	1d7f0
F	1d5d9	f	1d5f3	S	1d5e7	s	1d600	5	1d7f1
G	1d5da	g	1d5f4	T	1d5e8	t	1d601	6	1d7f2
H	1d5db	h	1d5f5	U	1d5e9	u	1d602	7	1d7f3
I	1d5dc	i	1d5f6	V	1d5ea	v	1d603	8	1d7f4
J	1d5dd	j	1d5f7	W	1d5ea	w	1d604	9	1d7f5
K	1d5de	k	1d5f8	X	1d5eb	x	1d605		
L	1d5df	l	1d5f9	Y	1d5ec	y	1d606		
M	1d5e1	m	1d5fa	Z	1d5ed	z	1d607		

Команда $\mathtt{\bullet}$ выводит буквы и цифры математического моноширинного шрифта, представленные в табл. В.4. Имеются только символы нормальной насыщенности.

Шрифт Typewriter

Таблица В.4

$\mathtt{\bullet}$									
•	U+#	•	U+#	•	U+#	•	U+#	•	U+#
A	1d670	a	1d68a	N	1d67d	n	1d697	0	1d7f6
B	1d671	b	1d68b	O	1d67e	o	1d698	1	1d7f7
C	1d672	c	1d68c	P	1d67f	p	1d699	2	1d7f8
D	1d673	d	1d68d	Q	1d680	q	1d69a	3	1d7f9
E	1d674	e	1d68e	R	1d681	r	1d69b	4	1d7fa
F	1d675	f	1d68f	S	1d682	s	1d69c	5	1d7fb
G	1d676	g	1d690	T	1d683	t	1d69d	6	1d7fc
H	1d677	h	1d691	U	1d684	u	1d69e	7	1d7fd
I	1d678	i	1d692	V	1d685	v	1d69f	8	1d7fe
j	1d693	J	1d679	W	1d686	w	1d6a0	9	1d7ff
K	1d67a	k	1d694	X	1d687	x	1d6a1		
L	1d67b	l	1d695	Y	1d688	y	1d6a2		
M	1d67c	m	1d696	Z	1d689	z	1d6a3		

Команда $\mathcal{\bullet}$ выводит прописные буквы каллиграфического шрифта, представленные в табл. В.5. На месте строчных букв и цифр стоят другие символы. Для вывода жирных букв используется команда $\boldsymbol{\mathcal{\bullet}}$. Символы шрифта отсутствуют в кодировке unicode и потому показаны списком.

Каллиграфический шрифт

Таблица В.5

$\mathcal{\bullet}$
<i>ABCDEFGHIJKLMN OPQRSTUVWXYZ</i>
$\boldsymbol{\mathcal{\bullet}}$
<i>ABCDEFGHIJKLMN OPQRSTUVWXYZ</i>

Пакет `eucal` из коллекции пакетов `amslatex` подключает рукописный шрифт. Загрузка пакета без параметров замещает каллиграфический шрифт рукописным. При загрузке с параметром `mathscr` доступны оба шрифта, так как для рукописного вводится новая команда `\mathscr{●}`.

Рукописный шрифт содержит только прописные буквы. Буквы «fgjnх» замещены символами «{}|\\$», остальные строчные буквы и цифры отсутствуют. Для вывода жирных символов используется команда `\boldsymbol{●}`.

Таблица В.6

Рукописный шрифт							
<code>\mathscr{●}</code>							
●	U+#	●	U+#	●	U+#	●	U+#
<i>A</i>	1d49c	<i>ℋ</i>	210b	<i>ℒ</i>	1d4aa	<i>ℳ</i>	1d4b1
<i>B</i>	212c	<i>ℐ</i>	2110	<i>ℙ</i>	1d4ab	<i>ℴ</i>	1d4b2
<i>C</i>	1d49e	<i>ℑ</i>	1d4a5	<i>ℚ</i>	1d4ac	<i>ℵ</i>	1d4b3
<i>D</i>	1d49f	<i>ℒ</i>	1d4a6	<i>ℛ</i>	211b	<i>ℶ</i>	1d4b4
<i>E</i>	2130	<i>ℓ</i>	2112	<i>℔</i>	1d4ae	<i>ℷ</i>	1d4b5
<i>F</i>	2131	<i>ℓ</i>	2133	<i>ℕ</i>	1d4af		
<i>G</i>	1d4a2	<i>ℓ</i>	1d4a9	<i>ℯ</i>	1d4b0		

<code>\boldsymbol{\mathscr{●}}</code>							
●	U+#	●	U+#	●	U+#	●	U+#
<i>A</i>	1d4d0	<i>ℋ</i>	1d4d7	<i>ℒ</i>	1d4de	<i>ℳ</i>	1d4e5
<i>B</i>	1d4d1	<i>ℐ</i>	1d4d8	<i>ℙ</i>	1d4df	<i>ℴ</i>	1d4e6
<i>C</i>	1d4d2	<i>ℑ</i>	1d4d9	<i>ℚ</i>	1d4e0	<i>ℵ</i>	1d4e7
<i>D</i>	1d4d3	<i>ℒ</i>	1d4da	<i>ℛ</i>	1d4e1	<i>ℶ</i>	1d4e8
<i>E</i>	1d4d4	<i>ℓ</i>	1d4db	<i>℔</i>	1d4e2	<i>ℷ</i>	1d4e9
<i>F</i>	1d4d5	<i>ℓ</i>	1d4dc	<i>ℕ</i>	1d4e3		
<i>G</i>	1d4d6	<i>ℓ</i>	1d4dd	<i>ℯ</i>	1d4e4		

Пакет `amssymb` из коллекции `amslatex`, автоматически загружаемый пакетом `amssymb`, подключает ажурный и готический шрифты.

\mathfrak{•}							
•	U+#	•	U+#	•	U+#	•	U+#
Ⓐ	1d504	ⓐ	1d51e	Ⓝ	1d511	Ⓢ	1d52b
Ⓑ	1d505	ⓑ	1d51f	Ⓞ	1d512	Ⓣ	1d52c
Ⓒ	1d520	ⓒ	212d	Ⓟ	1d513	Ⓤ	1d52d
Ⓓ	1d507	ⓓ	1d521	Ⓠ	1d514	Ⓡ	1d52e
Ⓔ	1d508	ⓔ	1d522	Ⓢ	1d515	Ⓥ	1d52f
Ⓕ	1d509	ⓕ	1d523	Ⓣ	1d516	Ⓦ	1d530
Ⓖ	1d50a	ⓖ	1d524	Ⓤ	1d517	Ⓧ	1d531
Ⓗ	210c	ⓗ	1d525	Ⓡ	1d518	Ⓨ	1d532
Ⓙ	2111	ⓙ	1d526	Ⓥ	1d519	Ⓩ	1d533
Ⓝ	1d50d	Ⓥ	1d527	Ⓦ	1d51a	ⓐ	1d534
Ⓞ	1d50e	Ⓧ	1d528	Ⓧ	1d51b	ⓑ	1d535
Ⓟ	1d50f	Ⓨ	1d529	Ⓨ	1d51c	Ⓝ	1d536
Ⓠ	1d510	Ⓩ	1d52a	Ⓩ	2128	Ⓞ	1d537

\boldsymbol{\mathfrak{•}}							
•	U+#	•	U+#	•	U+#	•	U+#
Ⓐ	1d56c	ⓐ	1d586	Ⓝ	1d579	Ⓢ	1d593
Ⓑ	1d56d	ⓑ	1d587	Ⓞ	1d57a	Ⓣ	1d594
Ⓒ	1d56e	ⓒ	1d588	Ⓟ	1d57b	Ⓤ	1d595
Ⓓ	1d56f	ⓓ	1d589	Ⓠ	1d57c	Ⓡ	1d596
Ⓔ	1d570	ⓔ	1d58a	Ⓢ	1d57d	Ⓥ	1d597
Ⓕ	1d571	ⓕ	1d58b	Ⓣ	1d57e	Ⓦ	1d598
Ⓖ	1d571	ⓖ	1d58c	Ⓤ	1d57f	Ⓧ	1d599
Ⓗ	1d573	ⓗ	1d58d	Ⓡ	1d580	Ⓨ	1d59a
Ⓙ	1d574	ⓙ	1d58e	Ⓥ	1d581	Ⓩ	1d59b
Ⓝ	1d575	Ⓥ	1d58f	Ⓦ	1d582	ⓐ	1d59c
Ⓞ	1d576	Ⓧ	1d590	Ⓧ	1d583	ⓑ	1d59d
Ⓟ	1d577	Ⓨ	1d591	Ⓨ	1d584	Ⓝ	1d59e
Ⓠ	1d578	Ⓩ	1d592	Ⓩ	1d585	Ⓞ	1d59f

Готический шрифт содержит прописные и строчные буквы, а также цифры нормальной и жирной насыщенности. Для вывода символов используются команды `\mathfrac{•}` и `\boldsymbol{•}`. Буквы готического шрифта представлены в табл. В.7 вместе шестнадцатеричными кодами кодировки `unicode`, а не входящие в нее цифры — без кодов.

Буквы ажурного шрифта, представленные в табл. В.8, выводит команда `\mathbb{•}`. В нем имеются только прописные буквы, а на месте строчных букв и цифр стоят другие символы. Жирный шрифт отсутствует.

Таблица В.8

Ажурный шрифт Double-struck (blackboard bold)

$\mathbb{•}$							
•	U+#	•	U+#	•	U+#	•	U+#
A	1d538	H	210d	O	1d546	V	1d54d
B	1d539	I	1d540	P	2119	W	1d54e
C	2102	J	1d541	Q	211a	X	1d54f
D	1d53b	K	1d542	R	211d	Y	1d550
E	1d53c	L	1d543	S	1d54a	Z	2124
F	1d53d	M	1d544	T	1d54b		
G	1d53e	N	2115	U	1d54c		

Команды расстановки математических акцентов представлены в табл. В.9. В колонках, помеченных значком •, показаны примеры акцентированных букв.

Таблица В.9

Математические акценты

•	Команды	•	Команды	•	Команды
ã	<code>\tilde{•}</code>	á	<code>\dot{•}</code>	ä	<code>\dddots{•}</code>
ā	<code>\bar{•}</code>	ä	<code>\ddots{•}</code>	ä	<code>\ddddots{•}</code>
â	<code>\hat{•}</code>	ǎ	<code>\check{•}</code>	abc	<code>\widehat{•}</code>
à	<code>\grave{•}</code>	á	<code>\acute{•}</code>	abc	<code>\widetilde{•}</code>
ă	<code>\breve{•}</code>	ā	<code>\vec{•}</code>		

В.2. Греческие буквы

В табл. В.10 приведены греческие буквы вместе со своими кодами в кодировке `unicode` и командами, используемыми для их обозначения в формулах.

Таблица В.10

Греческие буквы

•	U+#	Команды	•	U+#	Команды
α	3b1	<code>\alpha</code>	ν	3bd	<code>\nu</code>
β	3b2	<code>\beta</code>	ξ	3be	<code>\xi</code>
γ	3b3	<code>\gamma</code>	π	3c0	<code>\pi</code>
δ	3b4	<code>\delta</code>	ϖ	3d6	<code>\varpi</code>
ϵ	3f5	<code>\epsilon</code>	ρ	3c1	<code>\rho</code>
ε	3b5	<code>\varepsilon</code>	ϱ	3f1	<code>\varrho</code>
ζ	3b6	<code>\zeta</code>	σ	3c3	<code>\sigma</code>
η	3b7	<code>\eta</code>	ς	3c2	<code>\varsigma</code>
θ	3b8	<code>\theta</code>	τ	3c4	<code>\tau</code>
ϑ	3d1	<code>\vartheta</code>	υ	3c5	<code>\upsilon</code>
ι	3b9	<code>\iota</code>	ϕ	3d5	<code>\phi</code>
κ	3ba	<code>\kappa</code>	φ	3c6	<code>\varphi</code>
\varkappa	3f0	<code>\varkappa</code>	χ	3c7	<code>\chi</code>
λ	3bb	<code>\lambda</code>	ψ	3c8	<code>\psi</code>
μ	3bc	<code>\mu</code>	ω	3c9	<code>\omega</code>
\digamma	3dc	<code>\digamma</code>			
Γ	393	<code>\Gamma</code>	Γ	1d6e4	<code>\Gamma</code>
Σ	3a3	<code>\Sigma</code>	Σ	1d6f4	<code>\Sigma</code>
Δ	393	<code>\Delta</code>	Δ	1d6e5	<code>\Delta</code>
Υ	3a5	<code>\Upsilon</code>	Υ	1d6f6	<code>\Upsilon</code>
Θ	398	<code>\Theta</code>	Θ	1d6e9	<code>\Theta</code>
Φ	3a6	<code>\Phi</code>	Φ	1d6f7	<code>\Phi</code>
Ψ	3a8	<code>\Psi</code>	Ψ	1d6f9	<code>\Psi</code>
Ξ	39e	<code>\Xi</code>	Ξ	1d6ef	<code>\Xi</code>
Ω	3a9	<code>\Omega</code>	Ω	1d6fa	<code>\Omega</code>
Π	3a0	<code>\Pi</code>	Π	1d6f1	<code>\Pi</code>

В.3. Математические символы

В табл. В.11 представлены большие операторы, относящиеся к категории `\mathop`. Их размеры зависят от типа формулы. Первые символы в таблице соответствуют размеру операторов в строчных формулах, а вторые — в вынесенных формулах.

Таблица В.11

Большие операторы

Символы	U+#	Команды	Символы	U+#	Команды
$\int \int$	222b	<code>\int</code>	$\Sigma \Sigma$	2211	<code>\sum</code>
$\oint \oint$	222e	<code>\oint</code>	$\Pi \Pi$	220f	<code>\prod</code>
$\iint \iint$	222c	<code>\iint</code>	$\coprod \coprod$	2210	<code>\coprod</code>
$\iiint \iiint$	222d	<code>\iiint</code>	$\bigcap \bigcap$	22c2	<code>\bigcap</code>
$\iiint \iiint$	2a0c	<code>\iiiint</code>	$\bigcup \bigcup$	22c3	<code>\bigcup</code>
$\int \dots \int \int \dots \int$		<code>\idotsint</code>	$\biguplus \biguplus$	2a04	<code>\biguplus</code>
$\odot \odot$	2a00	<code>\bigodot</code>	$\bigsqcup \bigsqcup$	2a06	<code>\bigsqcup</code>
$\otimes \otimes$	2a02	<code>\bigotimes</code>	$\bigvee \bigvee$	22c1	<code>\bigvee</code>
$\oplus \oplus$	2a01	<code>\bigoplus</code>	$\bigwedge \bigwedge$	22c0	<code>\bigwedge</code>

Большие операторы могут иметь верхний и нижний пределы (индексы), положение которых регулируют команды `\limits`, `\nolimits`. С помощью команды

`\sideset{левые индексы}{правые индексы}`

пределы (индексы) можно поставить как слева, так и справа.

Табл. В.12 содержит разделители, размер которых может меняться в зависимости от высоты заключенного в них выражения. Левые и правые скобки, а также разделители с именами, начи-

нающимися с букв `l` и `r`, относятся к категориям `\mathopen` и `\mathclose`. Они имеют несимметричную отбивку слева и справа. Остальные разделители относятся к категории `\mathord`.

Разделители

Таблица В.12

•	U+#	Команды	•	U+#	Команды
(28	()	29)
[5b	[]	5d]
{	7b	<code>\{</code> , <code>\lbrace</code>	}	7d	<code>\}</code> , <code>\rbrace</code>
/	2f	/	\	5c	<code>\backslash</code>
⟨	27e8	<code>\langle</code>	⟩	27e9	<code>\rangle</code>
⌈	2308	<code>\lceil</code>	⌋	2309	<code>\rceil</code>
⌊	230a	<code>\lfloor</code>	⌋	230b	<code>\rfloor</code>
⌜	231c	<code>\ulcorner</code>	⌝	231d	<code>\urcorner</code>
⌞	231e	<code>\llcorner</code>	⌟	231f	<code>\lrcorner</code>
↑	2191	<code>\uparrow</code>	↓	2193	<code>\downarrow</code>
↕	2195	<code>\updownarrow</code>	↕	21d5	<code>\Updownarrow</code>
⇓	21d3	<code>\Downarrow</code>	⇑	21d1	<code>\Uparrow</code>
		<code>\lvert</code>			<code>\rvert</code>
		<code>\lVert</code>			<code>\rVert</code>
	7c	<code> </code> , <code>\vert</code>			

В табл. В.13–В.18 собраны различные символы, представленные вместе со своими кодами в кодировке `unicode` и командами, используемыми для их обозначения в формулах. Символы первых двух таблиц относятся к категории `\mathord`, третьей — к категории `\mathbin`, а остальных — к категории `\mathrel`.

Таблица В.13

Буквенные символы

•	U+#	Команды	•	U+#	Команды	•	U+#	Команды
ℑ	2111	\Im	℞	211c	\Re	ℵ	2135	\aleph
ℏ	210f	\hslash	ℏ		\hbar	⋈	2136	\beth
ι	1d6a4	\imath	⋈	1d6a5	\jmath	⋇	2137	\gimel
ℓ	1d55c	\Bbbk	ℓ	2113	\ell	⌈	2138	\daleth
⋈	2132	\Finv	⊃	2141	\Game	ø	f0	\eth
¥	a5	\yen	ℳ	2127	\mho	ϕ	2118	\wp

Таблица В.14

Различные символы

•	U+#	Команды	•	U+#	Команды
∞	221e	\infty	∇	2207	\nabla
∂	2202	\partial	∀	2200	\forall
∖		\diagdown	∕		\diagup
⊥	22a5	\bot	⊤	22a4	\top
∃	2203	\exists	∄	2204	\nexists
∖	2035	\backprime	/	2032	\prime
¬	ac	\neg, \lnot	∅	29b0	\emptyset
℄	2201	\complement	∅	2205	\varnothing
∠	2221	\measuredangle	∠	2220	\angle
∠	2222	\sphericalangle	√	221a	\surd
△	25b3	\triangle	∇	25bf	\triangledown
▼	25be	\blacktriangledown	▲	25b4	\blacktriangle
□	25a1	\square, \Box	■	25a0	\blacksquare
◇	25ca	\lozenge	◆	29eb	\blacklozenge
◇	2662	\diamondsuit	♥	2661	\heartsuit
♣	2663	\clubsuit	♠	2660	\spadesuit
★	2605	\bigstar	♣	2720	\maltese
♯	266f	\sharp	♭	266d	\flat
♮	266e	\natural	✓	2713	\checkmark
®	ae	\circledR	Ⓢ	24c8	\circledS

•	U+#	Команды	•	U+#	Команды
±	b1	<code>\pm</code>	∓	2213	<code>\mp</code>
×	d7	<code>\times</code>	⋈	2214	<code>\dotplus</code>
÷	f7	<code>\div</code>	⋆	22c7	<code>\divideontimes</code>
⋈	22cb	<code>\leftthreetimes</code>	⋈	22cc	<code>\rightthreetimes</code>
⋉	22c9	<code>\ltimes</code>	⋉	22ca	<code>\rtimes</code>
⋊	2216	<code>\smallsetminus</code>	⋋	29f5	<code>\setminus</code>
⋌	228e	<code>\uplus</code>	∏	2af3	<code>\amalg</code>
∩	2229	<code>\cap</code>	∪	222a	<code>\cup</code>
∩	2293	<code>\sqcap</code>	∩	2294	<code>\sqcup</code>
∩	22d2	<code>\Cap, \doublecap</code>	∪	22d3	<code>\Cup, \doublecup</code>
⋈	22d6	<code>\lessdot</code>	⋈	22d7	<code>\gtrdot</code>
∪	22ce	<code>\curlyvee</code>	∩	22cf	<code>\curlywedge</code>
∪	2228	<code>\vee, \lor</code>	∧	2227	<code>\wedge, \lnot</code>
∪	22bb	<code>\veebar</code>	∧	22bc	<code>\barwedge</code>
∪	22ba	<code>\intercal</code>	∧	2306	<code>\doublebarwedge</code>
†	2020	<code>\dagger</code>	†	2021	<code>\ddagger</code>
*	22c6	<code>\star</code>	*	204e	<code>\ast</code>
•	22c5	<code>\cdot</code>	•	2b1d	<code>\centerdot</code>
⊠	22a0	<code>\boxtimes</code>	⊠	229f	<code>\boxminus</code>
⊕	229e	<code>\boxplus</code>	⊠	22a1	<code>\boxdot</code>
∩	2240	<code>\wr</code>	•	2219	<code>\bullet</code>
○		<code>\bigcirc</code>	○	2218	<code>\circ</code>
⊘	2298	<code>\oslash</code>	⊘	229b	<code>\circledast</code>
⊗	2297	<code>\otimes</code>	⊖	2296	<code>\ominus</code>
⊕	2295	<code>\oplus</code>	⊙	2299	<code>\odot</code>
⊖	229d	<code>\circleddash</code>	⊙	229a	<code>\circledcirc</code>
▽	25bd	<code>\bigtriangledown</code>	△	25b3	<code>\bigtriangleup</code>
◁	25c3	<code>\triangleleft</code>	▷	26b9	<code>\triangleright</code>
◁	25c1	<code>\lhd</code>	▷	25b7	<code>\rhd</code>
◁	22b4	<code>\unlhd</code>	▷	22b5	<code>\unrhd</code>
◇	22c4	<code>\diamond</code>	◇	25c7	<code>\Diamond</code>

Далее следуют таблицы с символами соотношений и стрелок, относящимися к категории `\mathrel`.

Соотношения

Таблица В.16

•	U+#	Команды	•	U+#	Команды
\equiv	2261	<code>\equiv</code>	\cong	2245	<code>\cong</code>
\propto	221d	<code>\propto</code>	\varpropto		<code>\varpropto</code>
\sim	223c	<code>\sim</code>	\simeq	2243	<code>\simeq</code>
\approx	2248	<code>\approx</code>	\approxeq	224a	<code>\approxeq</code>
\backsimeq		<code>\backsimeq</code>	\backsim		<code>\backsim</code>
\thicksim		<code>\thicksim</code>	\thickapprox		<code>\thickapprox</code>
\eqcirc	2256	<code>\eqcirc</code>	\circeq	2257	<code>\circeq</code>
\doteq	2250	<code>\doteq</code>	\doteqdot , <code>\Doteq</code>	2251	<code>\doteqdot</code> , <code>\Doteq</code>
\fallingdotseq	2252	<code>\fallingdotseq</code>	\risingdotseq	2253	<code>\risingdotseq</code>
\Bumpeq	224e	<code>\Bumpeq</code>	\bumpeq	224f	<code>\bumpeq</code>
\in	2208	<code>\in</code>	\ni , <code>\owns</code>	220b	<code>\ni</code> , <code>\owns</code>
\ll	226a	<code>\ll</code>	\gg	226b	<code>\gg</code>
\lll , <code>\llless</code>	22d8	<code>\lll</code> , <code>\llless</code>	\ggg , <code>\gggtr</code>	22d9	<code>\ggg</code> , <code>\gggtr</code>
\leq , <code>\le</code>	2264	<code>\leq</code> , <code>\le</code>	\geq , <code>\ge</code>	2265	<code>\geq</code> , <code>\ge</code>
\leqq	2266	<code>\leqq</code>	\geqq	2267	<code>\geqq</code>
\leqslant	2a7d	<code>\leqslant</code>	\geqslant	2a7e	<code>\geqslant</code>
\eqslantless	2a95	<code>\eqslantless</code>	\eqslantgtr	2a96	<code>\eqslantgtr</code>
\lesssim	2272	<code>\lesssim</code>	\gtrsim	2273	<code>\gtrsim</code>
\lessapprox	2a85	<code>\lessapprox</code>	\gtrapprox	2a86	<code>\gtrapprox</code>
\lessgtr	2276	<code>\lessgtr</code>	\gtrless	2277	<code>\gtrless</code>
\lesseqgtr	22da	<code>\lesseqgtr</code>	\gtreqless	22db	<code>\gtreqless</code>
\lesseqqgtr	2a2b	<code>\lesseqqgtr</code>	\gtreqqlless	2a2c	<code>\gtreqqlless</code>
\subset	2282	<code>\subset</code>	\supset	2283	<code>\supset</code>
\subseteq	2286	<code>\subseteq</code>	\supseteq	2287	<code>\supseteq</code>
\subsetneq	2ac5	<code>\subsetneq</code>	\supsetneqq	2ac6	<code>\supsetneqq</code>
\Subset	22d0	<code>\Subset</code>	\Supset	22d1	<code>\Supset</code>
\sqsubset	228f	<code>\sqsubset</code>	\sqsupset	2290	<code>\sqsupset</code>
\sqsubseteq	2291	<code>\sqsubseteq</code>	\sqsupseteq	2292	<code>\sqsupseteq</code>

•	U+#	Команды	•	U+#	Команды
\prec	227a	<code>\prec</code>	\succ	227b	<code>\succ</code>
\preceq	2aaf	<code>\preceq</code>	\succeq	2ab0	<code>\succeq</code>
$\prec\curlyeq$	227c	<code>\prec\curlyeq</code>	$\succ\curlyeq$	227d	<code>\succ\curlyeq</code>
\curlyeqprec	22de	<code>\curlyeqprec</code>	\curlyeqsucc	22df	<code>\curlyeqsucc</code>
\precsim	227e	<code>\precsim</code>	\succsim	227f	<code>\succsim</code>
\precapprox	2ab7	<code>\precapprox</code>	\succapprox	2ab8	<code>\succapprox</code>
\vartriangleleft	22b2	<code>\vartriangleleft</code>	\trianglerightleftarrow	22b4	<code>\trianglerightleftarrow</code>
\vartriangleright	22b3	<code>\vartriangleright</code>	\triangleangleright	22b5	<code>\triangleangleright</code>
\blacktriangleleft	25c2	<code>\blacktriangleleft</code>	\bowtie	22c8	<code>\bowtie</code>
\blacktriangleright	25b8	<code>\blacktriangleright</code>	\Join	2a1d	<code>\Join</code>
\triangle	25b5	<code>\vartriangle</code>	\triangleq	225c	<code>\triangleq</code>
\therefore	2234	<code>\therefore</code>	\because	2235	<code>\because</code>
\frown	2222	<code>\frown</code>	\smile	2223	<code>\smile</code>
\smallfrown		<code>\smallfrown</code>	\smallsmile		<code>\smallsmile</code>
\between	226c	<code>\between</code>	\asymp	224d	<code>\asymp</code>
\mid	2223	<code>\mid</code>	\shortmid		<code>\shortmid</code>
\parallel	2225	<code>\parallel</code>	\shortparallel		<code>\shortparallel</code>
\vdash	22a2	<code>\vdash</code>	\dashv	22a3	<code>\dashv</code>
\Vdash	22A7	<code>\models</code>	\VDash	22a8	<code>\vDash</code>
\Vdash	22a9	<code>\Vdash</code>	\Vvdash	22aa	<code>\Vvdash</code>
\perp	27c2	<code>\perp</code>	\pitchfork	22d4	<code>\pitchfork</code>
\backepsilon	3f6	<code>\backepsilon</code>			

Таблица В.17

Стрелки (соотношения)

•	U+#	Команды	•	U+#	Команды
\leftarrow	2190	<code>\leftarrow, \gets</code>	\rightarrow	2192	<code>\rightarrow, \to</code>
\nleftarrow	219a	<code>\nleftarrow</code>	\nrightrightarrow	219b	<code>\nrightrightarrow</code>
\longleftarrow	27f5	<code>\longleftarrow</code>	\longrightarrow	27f6	<code>\longrightarrow</code>
\leftrightsquigarrow	21c7	<code>\leftrightsquigarrow</code>	\rightleftarrows	21c6	<code>\leftrightsquigarrow</code>
\rightrightarrows	21c9	<code>\rightrightarrows</code>	\rightleftarrows	21c4	<code>\rightleftarrows</code>

•	U+#	Команды	•	U+#	Команды
\leftarrow	21d0	<code>\Leftarrow</code>	\Rightarrow	21d2	<code>\Rightarrow</code>
\nleftarrow	21cd	<code>\nLeftarrow</code>	\nrightarrow	21cf	<code>\nrightarrow</code>
\Longleftarrow	27f8	<code>\Longleftarrow</code>	\Longrightarrow	27f9	<code>\Longrightarrow</code>
\Lleftarrow	21da	<code>\Lleftarrow</code>	\Rrightarrow	21db	<code>\Rrightarrow</code>
\leftrightarrow	2194	<code>\leftrightarrow</code>	\leftrightarrow	21d4	<code>\leftrightarrow</code>
\longleftrightarrow	27f7	<code>\longleftrightarrow</code>	\leftrightsquigarrow	21ae	<code>\leftrightsquigarrow</code>
\longleftrightarrow	27fa	<code>\longleftrightarrow</code>	\nleftrightarrow	21ce	<code>\nleftrightarrow</code>
\uparrow	2191	<code>\uparrow</code>	\downarrow	2193	<code>\downarrow</code>
\Uparrow	21c8	<code>\Uparrow</code>	\Downarrow	21ca	<code>\Downarrow</code>
\Updownarrow	21d1	<code>\Updownarrow</code>	\Downarrow	21d3	<code>\Downarrow</code>
\updownarrow	2195	<code>\updownarrow</code>	\Updownarrow	21d5	<code>\Updownarrow</code>
\nearrow	2197	<code>\nearrow</code>	\searrow	2198	<code>\searrow</code>
\nwarrow	2196	<code>\nwarrow</code>	\swarrow	2199	<code>\swarrow</code>
\leftharpoonright	21bd	<code>\leftharpoonright</code>	\leftarrow	21bc	<code>\leftarrow</code>
\rightarrow	21c1	<code>\rightarrow</code>	\rightarrow	21c0	<code>\rightarrow</code>
\leftrightharpoons	21cb	<code>\leftrightharpoons</code>	\upharpoonleft	21bf	<code>\upharpoonleft</code>
\rightleftharpoons	21cc	<code>\rightleftharpoons</code>	\downharpoonleft	21c3	<code>\downharpoonleft</code>
\downharpoonright	21c2	<code>\downharpoonright</code>	\upharpoonright	21be	<code>\upharpoonright</code>
					<code>\restriction</code>
\mapsto	21a6	<code>\mapsto</code>	\longmapsto	27fc	<code>\longmapsto</code>
\multimap	22b8	<code>\multimap</code>	\leadsto	2933	<code>\leadsto</code>
\leftrightsquigarrow	21ad	<code>\leftrightsquigarrow</code>	\rightsquigarrow	21dd	<code>\rightsquigarrow</code>
\dashleftarrow	21e0	<code>\dashleftarrow</code>	\dashrightarrow	21e2	<code>\dashrightarrow</code>
\twoheadleftarrow	219e	<code>\twoheadleftarrow</code>	\leftarrowtail	21a2	<code>\leftarrowtail</code>
\twoheadrightarrow	21a0	<code>\twoheadrightarrow</code>	\rightarrowtail	21a3	<code>\rightarrowtail</code>
\hookrightarrow	21a9	<code>\hookrightarrow</code>	\hookrightarrow	21aa	<code>\hookrightarrow</code>
\looparrowleft	21ab	<code>\looparrowleft</code>	\looparrowright	21ac	<code>\looparrowright</code>
\curvearrowleft	21b6	<code>\curvearrowleft</code>	\curvearrowright	21b7	<code>\curvearrowright</code>
\circlearrowleft	21ba	<code>\circlearrowleft</code>	\Lsh	21b0	<code>\Lsh</code>
\circlearrowright	21bb	<code>\circlearrowright</code>	\Rsh	21b1	<code>\Rsh</code>

Соотношения с отрицанием

•	U+#	Команды	•	U+#	Команды
\neq	2260	<code>\neq, \ne</code>	\ncong	2245	<code>\ncong</code>
\nsim	2241	<code>\nsim</code>	\ngtr	226f	<code>\ngtr</code>
\nless	226e	<code>\nless</code>	\ngeq	2271	<code>\ngeq</code>
\nleq	2270	<code>\nleq</code>	\gneq	2a88	<code>\gneq</code>
\nleqslant	2a87	<code>\nleqslant</code>	\ngeqslant		<code>\ngeqslant</code>
\nleqq		<code>\nleqq</code>	\ngeqq		<code>\ngeqq</code>
\lvertneqq	2268	<code>\lvertneqq</code>	\gneqq	2269	<code>\gneqq</code>
\lnsim	22e6	<code>\lnsim</code>	\gvertneqq		<code>\gvertneqq</code>
\lnapprox	2a89	<code>\lnapprox</code>	\gnsim	22e7	<code>\gnsim</code>
\nsubseteq	2288	<code>\nsubseteq</code>	\nsupseteq	2289	<code>\nsupseteq</code>
\subsetneq	228a	<code>\subsetneq</code>	\supsetneq	228b	<code>\supsetneq</code>
\varsubsetneq		<code>\varsubsetneq</code>	\varsupsetneq		<code>\varsupsetneq</code>
\nsubseteqq		<code>\nsubseteqq</code>	\nsupseteqq		<code>\nsupseteqq</code>
\subsetneqq	2acb	<code>\subsetneqq</code>	\supsetneqq	2acc	<code>\supsetneqq</code>
\varsubsetneqq		<code>\varsubsetneqq</code>	\varsupsetneqq		<code>\varsupsetneqq</code>
\nprec	2280	<code>\nprec</code>	\nsucc	2281	<code>\nsucc</code>
\npreceq		<code>\npreceq</code>	\nsucceq		<code>\nsucceq</code>
\precneqq	2ab5	<code>\precneqq</code>	\succneqq	2ab6	<code>\succneqq</code>
\precnsim	22e8	<code>\precnsim</code>	\succnsim	22e9	<code>\succnsim</code>
\precnapprox	2ab9	<code>\precnapprox</code>	\succnapprox	2aba	<code>\succnapprox</code>
\ntriangleleft	22ea	<code>\ntriangleleft</code>	\ntriangleright	22eb	<code>\ntriangleright</code>
\ntrianglelefteq	22ec	<code>\ntrianglelefteq</code>	\ntrianglerighteq	22ed	<code>\ntrianglerighteq</code>
\nmid	2224	<code>\nmid</code>	\nparallel	2226	<code>\nparallel</code>
\nshortmid		<code>\nshortmid</code>	\nshortparallel		<code>\nshortparallel</code>
\nvdash	22ac	<code>\nvdash</code>	\nVDash	22ad	<code>\nVDash</code>
\nVdash	22ae	<code>\nVdash</code>	\nVDash	22af	<code>\nVDash</code>

Отрицание отсутствующих в таблице отношений можно построить с помощью оператора `\not`, например:

$$\not\prec\curvearrowright \rightsquigarrow \nprec.$$

В.4. Точки и многоточия

В табл. В.19 представлены различные виды точек и многоточий, относящиеся к категории `\mathpunct`. Для символов из первой колонки, входящих в кодировку `unicode`, приведены шестнадцатеричные коды.

Точки и многоточия

Таблица В.19

Примеры	U+#	Команды	Примеры	Команды
$x : x$	2236	<code>\colon</code>	$x . x$	<code>\ldotp</code>
$x \cdot x$	b7	<code>\cdot</code>	$x \cdot x$	<code>\cdotp</code>
$x \dots x$	2026	<code>\ldots</code>	$0 \dots 9$	<code>\dotso</code>
$x \cdots x$	22ef	<code>\cdots</code>	$+ \cdots$	<code>\dotsb</code>
$x \vdots x$	22ee	<code>\vdots</code>	$\times \cdots$	<code>\dotism</code>
$x \ddots x$	22f1	<code>\ddots</code>	$x \dots x$	<code>\dots</code>
$\int_b^a \dots$		<code>\dotssi</code>	$, \dots$	<code>\dotsc</code>

Символы `\cdotp`, `\ldotp` и `\colon` имеют правую отбивку. Отбивка обычного двоеточия симметрична, а точка отбивается слева. Сравните:

$$x . x \rightsquigarrow x . x \quad \text{и} \quad x : x \leftarrow x \ldotp x$$

или

$$x : x \rightsquigarrow x : x \quad \text{и} \quad x : x \leftarrow x \colon x$$

Многоточия `\dotsc`, `\dotsc`, `\dotssi`, `\dotism` и `\dotso` предназначены для использования, соответственно, после бинарного оператора, запятой, интеграла, знака умножения и чего-то другого.

Положение многоточия, заданного командой `\dots`, изменяется в зависимости от контекста выражения, следующего за ним.

В.5. Математические функции

В табл. В.20 и В.21 приведены стандартные функции и пределы, определенные при загрузке настроек русского языка в пакете `babel` и использовании пакета `amsmath`.

Математические функции

Таблица В.20

Примеры	Команды	Примеры	Команды
$\sin x$	<code>\sin</code>	$\arcsin x$	<code>\arcsin</code>
$\cos x$	<code>\cos</code>	$\arccos x$	<code>\arccos</code>
$\tan x$	<code>\tan</code>	$\arctan x$	<code>\arctan</code>
$\operatorname{tg} x$	<code>\operatorname{tg}</code>	$\operatorname{arctg} x$	<code>\operatorname{arctg}</code>
$\operatorname{ctg} x$	<code>\operatorname{ctg}</code>	$\operatorname{arcctg} x$	<code>\operatorname{arcctg}</code>
$\operatorname{cot} x$	<code>\cot</code>	$\operatorname{cosec} x$	<code>\operatorname{cosec}</code>
$\operatorname{sec} x$	<code>\sec</code>	$\operatorname{csc} x$	<code>\operatorname{csc}</code>
$\sinh x$	<code>\sinh</code>	$\operatorname{sh} x$	<code>\operatorname{sh}</code>
$\cosh x$	<code>\cosh</code>	$\operatorname{ch} x$	<code>\operatorname{ch}</code>
$\tanh x$	<code>\tanh</code>	$\operatorname{th} x$	<code>\operatorname{th}</code>
$\operatorname{coth} x$	<code>\operatorname{coth}</code>	$\operatorname{cth} x$	<code>\operatorname{cth}</code>
$\exp x$	<code>\exp</code>	$\ln x$	<code>\ln</code>
$\log x$	<code>\log</code>	$\lg x$	<code>\lg</code>
$\arg x$	<code>\arg</code>	$\operatorname{hom} x$	<code>\operatorname{hom}</code>
$\operatorname{deg} x$	<code>\operatorname{deg}</code>	$\operatorname{ker} x$	<code>\operatorname{ker}</code>
$\operatorname{dim} x$	<code>\operatorname{dim}</code>	$\operatorname{sup} x$	<code>\operatorname{sup}</code>
$\operatorname{D} x$	<code>\operatorname{Variance}</code>	$\operatorname{P} x$	<code>\operatorname{Prob}</code>
$y \bmod x$	<code>\bmod</code>	$y \operatorname{mod} x$	<code>\operatorname{mod}</code>
$y \pmod{x}$	<code>\pmod</code>	$y(x)$	<code>\operatorname{pod}</code>

Функции, представленные в табл. В.21, могут иметь верхние и нижние индексы, которые регулируются так же, как пределы больших операторов.

Функции, имеющие пределы

Примеры	Команды	Примеры	Команды
$\det \dots$	<code>\det</code>	$\gcd \dots$	<code>\gcd</code>
$\inf \dots$	<code>\inf</code>	$\Pr \dots$	<code>\Pr</code>
$\max \dots$	<code>\max</code>	$\min \dots$	<code>\min</code>
$\lim \dots$	<code>\lim</code>		
$\limsup \dots$	<code>\limsup</code>	$\overline{\lim} \dots$	<code>\varlimsup</code>
$\liminf \dots$	<code>\liminf</code>	$\underline{\lim} \dots$	<code>\varliminf</code>
$\inj \lim \dots$	<code>\injlim</code>	$\lim_{\rightarrow} \dots$	<code>\varinjlim</code>
$\proj \lim \dots$	<code>\projlim</code>	$\lim_{\leftarrow} \dots$	<code>\varprojlim</code>

Имеется два способа создания функций, отсутствующих среди стандартных. Команды

$$\backslash\operatornamename{\{функция\}},$$

$$\backslash\operatornamename*{\{функция\}}$$

создают функции непосредственно при их использовании.

Команды

$$\backslash\DeclareMathOperator{\{команда\}}{\{функция\}}$$

$$\backslash\DeclareMathOperator*{\{команда\}}{\{функция\}}$$

определяют новую команду для вывода функции.

Команды со звездочкой в имени предназначены для создания функций, имеющих пределы.

Приложение Г

Генерация предметного указателя

Программа `texindy` запускает и настраивает программу `xindy` для генерации указателя. Настройки разбиты на большое число файлов, входящих в состав пакета `xindy`. Они имеют расширение `xdu` и представляют собой части кода, написанного на языке Lisp.

В простейшем случае при запуске `texindy` необходимо задать язык и кодировку записей. Приведем пример генерации русского указателя в кодировке `utf8` из записей, находящихся в файле `•.idx`:¹

```
texindy.exe -C utf8 -L russian •.idx
```

-C — ключ загрузки кодировки,

-L — ключ загрузки языка.

Вместо `•` нужно подставить имя файла, а сверстанный указатель будет записан в файл с тем же именем и расширением `•.ind`.

Правила сортировки русского текста добавляются к загружаемым по умолчанию правилам сортировки латинских букв,

¹ В программе TeXstudio командную строку `texindy` можно отредактировать в меню «Options ⇒ Configure TeXstudio... ⇒ Commands ⇒ Texindy».

поэтому русские и английские термины сортируются корректно. В то же время, русская часть указателя будет разбита на рубрики, а английские термины будут выведены общим списком. Для генерации рубрик в обеих частях указателя нужно составить и загрузить общую таблицу сортировки:

Файл rueng.xdy

```
;; порядок сортировки
(define-letter-groups ("A" "B" "C" "D" "E" "F" "G"
  "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
  "T" "U" "V" "W" "X" "Y" "Z"
  "А" "Б" "В" "Г" "Д" "Е" "Ж" "З" "И" "Й" "К" "Л"
  "М" "Н" "О" "П" "Р" "С" "Т" "У" "Ф" "Х" "Ц" "Ч"
  "Ш" "Щ" "Ъ" "Ы" "Ь" "Э" "Ю" "Я"))

;; общий алфавит
(define-rule-set "alphabetize"
  :rules (("a" "A") ("б" "Б") ("в" "В") ("г" "Г")
    ("д" "Д") ("е" "Е") ("ё" "Е") ("ё" "Е") ("ж" "Ж")
    ("з" "З") ("и" "И") ("й" "Й") ("к" "К") ("л" "Л")
    ("м" "М") ("н" "Н") ("о" "О") ("п" "П") ("р" "Р")
    ("с" "С") ("т" "Т") ("у" "У") ("ф" "Ф") ("х" "Х")
    ("ц" "Ц") ("ч" "Ч") ("ш" "Ш") ("щ" "Щ") ("ъ" "Ъ")
    ("ы" "Ы") ("ь" "Ь") ("э" "Э") ("ю" "Ю") ("я" "Я")
    ("a" "A") ("b" "B") ("c" "C") ("d" "D") ("e" "E")
    ("f" "F") ("g" "G") ("h" "H") ("i" "I") ("j" "J")
    ("k" "K") ("l" "L") ("m" "M") ("n" "N") ("o" "O")
    ("p" "P") ("q" "Q") ("r" "R") ("s" "S") ("t" "T")
    ("u" "U") ("v" "V") ("w" "W") ("x" "X") ("y" "Y")
    ("z" "Z")))

;; ë = e
(define-rule-set "reorderyo"
  :rules (("E" "8") ("e" "8") ("Ë" "9") ("ë" "9")))

;; специальные символы
```

```
(define-rule-set "ignore-special"
  :rules (("." "" :string) ("?" "" :string)
    ("!" "" :string) ("'" "" :string) ("-" "" :string)
    ("{" "" :string) ("}" "" :string)))
;; циклы сортировки
(use-rule-set :run 0
  :rule-set ("alphabetize" "ignore-special"))
(use-rule-set :run 1
  :rule-set ("reorder" "ignore-special"))
;; подчеркивание номеров страниц
(define-attributes (( "underline")) )
(markup-locref :open "\underline{" :close "}"
  :attr "underline")
```

Мы привели настройку сортировки англо-русского указателя, которую можно скопировать в файл, поместив его в папку с компилируемым документом или другой каталог, содержащий ресурсы, доступные компилирующим программам. Помимо правил сортировки в нее добавлена возможность подчеркивания страниц в указателе с помощью команд `\index{...|underline}`.

Настройки пользователя загружаются с помощью ключа `-M`. Следующая командная строка запускает `texindy` для сортировки указателя с настройками, находящимися в файле `rueng.xdy`:

```
texindy.exe -C utf8 -L russian -M rueng ●.idx
-M — ключ загрузки настроек пользователя.
```

Ключ `-M` можно использовать многократно, что позволяет комбинировать настройки, разбив их на несколько файлов.

Пакет `hyperref` позволяет преобразовать ссылки указателя в гиперссылки. Для этого определена команда

```
\hyperpage{номер страницы},
```

аргументом которой является номер страницы. Ее вывод можно оформить следующим образом:

```
;; оформление гиперссылок на номера страниц
(define-attributes (( "underline" ) )
(markup-locref :open "\underline{\hyperpage{":close "}}"
:attr "underline")
(markup-locref :open "\textbf{\hyperpage{":close "}}"
:attr "textbf")
(markup-locref :open "\textit{\hyperpage{":close "}}"
:attr "textit")
(markup-locref :open "\hyperpage{":close "}"
:attr "default")
```

Настройки, находящиеся в файле `hyperref.xdy`, позволяют выделить гиперссылки курсивом, жирным шрифтом и подчеркиванием. Они загружаются следующей командной строкой:

```
texindy -C utf8 -L russian -M rueng -M hyperref •.idx
```

Программу `texindy` можно настроить на верстку глоссария и списка аббревиатур. Для этого нужно восстановить двухстороннее выравнивание текста, так как по умолчанию в указателях он выравнивается только слева, подавить вывод запятой и номера страницы в конце записей и убрать буквы-заголовки секций указателя, излишние в глоссарии.

Требуемые настройки сведены в файл `glossary.xdy`, который следует загружать для генерации глоссария:

```
texindy -C utf8 -L russian -M rueng -M glossary •.idx
```

Если указатель и глоссарий формируются с помощью пакета `makeidx` из записей `\index` и `\glossary`, файлы `\.idx` и `\.glo` будут иметь одинаковое имя с разным расширением, поэтому `texindy` будет записывать результаты их сортировки в один файл. В этом случае, воспользовавшись ключом `-o`, для записи глоссария нужно указать файл с другим именем. Например, для генерации глоссария, сохраняемого в файле `glossary.tex`, следует использовать вызов:

```
texindy -Cutf8 -Lrussian -Mrueng -Mglossary -oglossary.tex •.glo
```

```

;; формирование окружения theindex
(markup-index :open
  "\begin{theindex}
% Убираем заголовки секций указателя
  \def\lettergroupDefault#1{
  \def\lettergroup#1{
% Убираем вертикальный пробел между секциями указателя
  %\let\indexspace\relax
% Восстанавливаем двухстороннее выравнивание
  \makeatletter
  \let\\\@centercr
  \@rightskip\z@skip \rightskip\z@skip
  \leftskip\z@skip\parindent\z@
  \makeatother
% Определяем команду, подавляющую вывод номера страницы
  \def\FakePageNumber#1{"
  :close "~n\end{description}~n"
  :tree)

;; подавление запятой в конце записи
(markup-locclass-list :open :sep )
;; подавление вывода номера страницы
(markup-locref :open "\FakePageNumber{":close "}")

```

Чтобы избежать коллизии с контролем команд ЛАТ_EX, в коде glossary.xdy использованы методы определения команд, применяемые непосредственно в Т_EX [1].

В настройке предусмотрена возможность изъять вертикальный пробел между группами терминов, начинающихся с разных букв. Для этого уберите комментарий перед строкой

```
\let\indexspace\relax.
```

Однако лучше использовать эти команды не на стадии генерации глоссария, а непосредственно перед его загрузкой в документ.

Список литературы и интернет-источников¹

1. Кнут Дональд Е. Все про \TeX . — Протвино: АО RDT \TeX , 1993. — 575 с. — ISBN 5-900614-01-8.
2. Львовский С.М. Набор и верстка в пакете \LaTeX . — М.: МЦНМО, 2014. — 398 с. — ISBN 978-5-4439-0239-5.
3. Котельников И., Чеботаев П. Настольная издательская система $\LaTeX 2_{\epsilon}$ по-русски. — Новосибирск: Сибирский хронограф, 2004. — 496 с. — ISBN 5-87550-195-2.
4. Гуссенс М., Миттельбах Ф., Самарин А. Путеводитель по пакету \LaTeX и его расширению $\LaTeX 2_{\epsilon}$. — М.: Мир, 1999. — 606 с. — ISBN 5-03-003325-4.
5. Гуссенс М., Ратц С. Путеводитель по пакету \LaTeX и его web-приложениям. — М.: Мир, 2001. — 604 с. — ISBN 5-03-003387-4.
6. Гуссенс М., Ратц С., Миттельбах Ф. Путеводитель по пакету \LaTeX и его графическим расширениям. — М.: Мир, 2002. — 621 с. — ISBN 5-03-003388-2.
7. Грэтцер Г. Первые шаги в \LaTeX 'е. — М.: Мир, 2000. — 172 с. — ISBN 5-03-003366-1.

¹ Данные об интернет-источниках, расположенных на официальном сайте \TeX -ресурсов: www.ctan.org, приведены по состоянию на 31.12.2020. Источники, содержащие этот адрес, являются электронными ресурсами.

8. Роженко А.И. Искусство верстки в Л^AT_EX'е. — Новосибирск: ИВМиМГ СО РАН, 2005. — 398 с. — ISBN 5-901548-25-6.
9. Балдин Е.М. Компьютерная типография Л^AT_EX. — СПб.: БХВ-Петербург, 2008. — 304 с. — ISBN 978-5-9775-0230-6.
10. Беляков С.Н., Палаш В.Н., Садовский П.А. Т_EX для всех: Оформление учебных и научных работ в системе Л^AT_EX. — М.: Книжный дом «ЛИБРОКОМ», 2009. — 208 с. — ISBN 978-5-397-00650-7.
11. REVTeX4.2 Author's Guide, 2018. — v.4.2c. — www.ctan.org/pkg/revtex.
12. elsarticle.cls – A better way to format your document, 2019. — v.3.2. — www.ctan.org/pkg/elsarticle.
13. Carlisle D. P. — Packages in the 'graphics' bundle, 2020. — 2020-08-21. — www.ctan.org/pkg/latex-graphics.
14. Fear Simon. — Publication quality tables in Л^AT_EX, 2016. — v.1.618033. — www.ctan.org/pkg/booktabs.
15. van Oostrum Piet, Øystein Bache, Leichter Jerry. — The multirow, bigstrut and bigdelim packages, 2016. — v.2.2. — www.ctan.org/pkg/multirow.
16. Mittelbach Frank. — An Extension of the Л^AT_EX theorem environment, 2014. — v2.2c. — www.ctan.org/pkg/theorem.
17. Rose Kristoffer H. — XY-pic User's Guide, 2013. — Version 3.8.9. — www.ctan.org/pkg/xypic.
18. Arseneau Donald. — The cite package: well formed numeric citations, 2015. — www.ctan.org/pkg/cite.
19. Kotelnikov Igor A. — The GOST package, 2017/01/12, v.1.2i edition, 2017. — www.ctan.org/pkg/gost.

20. Braams Johannes, Carlisle David, Jeffrey Alan et al. — The L^AT_EX₂e Sources, 2020. — 2020-02-02 Patch level 5. — www.ctan.org/pkg/source2e.
21. Thorup Kresten Krab, Jensen Frank, (and Chris Rowley). — The calc package. Infix notation arithmetic in L^AT_EX, 2020. — v4.3. — www.ctan.org/pkg/calc.
22. Carlisle David. — The color package, 2019. — 2019/11/23. — www.ctan.org/pkg/color.
23. Mittelbach Frank, Carlisle David. — A new implementation of L^AT_EX's tabular and array environment, 2019. — v2.4l. — www.ctan.org/pkg/array.
24. Carlisle David. — The dcolumn package, 2014. — v1.06. — www.ctan.org/pkg/dcolumn.
25. Carlisle David. — The hline package, 2020. — v2.04. — www.ctan.org/pkg/hline.
26. Carlisle David. — The longtable package, 2020. — v4.13. — www.ctan.org/pkg/longtable.
27. Carlisle David. — The tabularx package, 2020. — v2.1.1c. — www.ctan.org/pkg/tabularx.
28. Carlisle David. — The colortbl package, 2018. — v1.0d. — www.ctan.org/pkg/colortbl.
29. Arseneau Donald. — chapterbib. multiple bibliographies in L^AT_EX, 2010. — — www.ctan.org/pkg/chapterbib.
30. Carlisle David. — The afterpage package, 2014. — 2014/10/28. — www.ctan.org/pkg/afterpage.
31. van Oostrum Piet. — The fancyhdr and extramarks packages, 2019. — v.3.10. — www.ctan.org/pkg/fancyhdr.

32. Sommerfeldt Axel. — Customizing captions of floating environments, 2011. — v.3.0. — www.ctan.org/pkg/caption.
33. Daly Patrick W. — Natural Sciences Citations and References, 2010. — v.8.31b. — <https://www.ctan.org/pkg/natbib>.
34. Lehman Philipp with, Kime Philip, Wemheuer Moritz et al. — The `biblatex` Package. Programmable Bibliographies and Citations, 2018. — v.3.12. — www.ctan.org/pkg/biblatex.
35. Gregorio Enrico. — The package `imakeidx`, 2016. — v.1.3e. — www.ctan.org/pkg/imakeidx.
36. Talbot Nicola L. C. — User Manual for `glossaries.sty` v4.46, 2020. — v4.46. — www.ctan.org/pkg/glossaries.
37. Talbot Nicola L. C. — The `glossaries` package v4.46: a guide for beginners, 2020. — v4.46. — www.ctan.org/pkg/glossaries.
38. Carlisle David, Mittelbach Frank. — The `bm` package, 2019. — v1.2d. — www.ctan.org/pkg/bm.
39. Carlisle David. — The `delarray` package, 2014. — v1.01. — www.ctan.org/pkg/delarray.
40. Carlisle David. — The `enumerate` package, 2015. — v3.00. — www.ctan.org/pkg/enumerate.
41. Mittelbach Frank. — Footnotes in a multi-column layout, 2020. — v1.1f. — www.ctan.org/pkg/ftnright.
42. Mittelbach Frank. — An environment for multicolumn output, 2019. — v1.8y. — www.ctan.org/pkg/multicol.
43. Mittelbach Frank. — The `varioref` package, 2020. — v1.6c. — www.ctan.org/pkg/varioref.
44. Schöpf Rainer, Raichle Bernd, Rowley Chris. — A New Implementation of \LaTeX 's `verbatim` and `verbatim*` Environments, 2001. — 2001/03/12. — www.ctan.org/pkg/verbatim.

45. Carlisle David, Høgholm Morten. — The `xspace` package, 2014. — v1.13. — www.ctan.org/pkg/xspace.
46. Rahtz Sebastian, Oberdiek Heiko, Project The L^AT_EX3. — Hyper-text marks in L^AT_EX: a manual for `hyperref`, 2020. — v7.00e. — www.ctan.org/pkg/hyperref.
47. Carlisle David. — The `ifthen` package, 2014. — v1.1c. — www.ctan.org/pkg/ifthen.
48. Oberdiek Heiko. — The `epstopdf` package, 2020. — v2.11. — www.ctan.org/pkg/epstopdf-pkg.
49. Oberdiek Heiko. — The `pdflscape` package, 2019. — v0.12. — www.ctan.org/pkg/pdflscape.
50. Schlicht R. — The `microtype` package, 2019. — v2.7d. — www.ctan.org/pkg/microtype.
51. Robertson Will, Hosny Khaled, Gesang Philipp et al. — The `fontspec` package. Font selection for X_ƎL^AT_EX and LuaL^AT_EX, 2020. — v2.7i. — www.ctan.org/pkg/fontspec.
52. Robertson Will, Stephani Philipp, Wright Joseph et al. — Experimental Unicode mathematical typesetting: The `unicode-math` package, 2020. — 0.8q. — www.ctan.org/pkg/unicode-math.

Предметный указатель

Для удобства работы указатель разбит на несколько частей. Команды вывода многочисленных символов собраны в таблицах приложений:

текст

символы	290	бинарные операторы	313
буквы латиницы	293	большие операторы	310
буквы кириллицы	296	соотношения	314

формулы

греческие буквы	309	стрелки	315
функции	319	разделители	311
символы	312	точки и многоточия	318

В отдельные указатели сведены:

список окружений	344	список пакетов	350
список длин и счетчиков	346	список литературы	352

Команды с именами `\name` и `\name*` и одинаковым синтаксисом представлены общим именем `\name*`. Символы `{•}` и `[•]` обозначают аргументы и параметры.

<code>\</code> , 19	<code>--~</code> (—) (babel [russian]), 64
<code>_</code> , 19	<code>\"•</code> (ä), 60, 292
<code>--</code> (—), 63	<code>#</code> , 19
<code>---</code> (—), 63	<code>\#</code> , 19
<code>''</code> ("), 63	<code>\$</code> , 19, 88
<code>!{•}</code> (разделитель, array), 176	<code>\\$</code> , 19
<code>* [•]</code> , 41, 47	<code>\$\$</code> , 89
<code>\!</code> , 42	<code>%</code> , 19

$\%$, 19
 $\&$, 19
 $\&$, 19
 \langle , \rangle , 88
 \langle , \rangle (ifthen), 265
 \cdot (\acute{a}), 60, 292
 \prime (\acute{a}), 60, 292
 $\hat{\cdot}$ (\hat{a}), 60, 292
 $\tilde{\cdot}$ (\tilde{a}), 60, 292
 $\ast\{\bullet\}\{\bullet\}$, 175
 $\ast\{\bullet\}\{\bullet\}$ (hhline), 186
 \ast (multirow), 81
 \backslash , , 42
 \backslash , , 64
 \backslash , (\backslash), 63
 $\backslash-$, 40
 $\backslash\{$, $\backslash\}$, 19
 $\backslash:$, 42
 $\backslash;$, 42
 \ll (\ll) (babel [russian]), 63
 $\lt\{\bullet\}$ (вставка, array), 176
 \ll (\ll), 63
 $=$ (multirow), 81
 $\backslash=\bullet$ (\bar{a}), 60, 292
 \gg (\gg) (babel [russian]), 63
 $\gt\{\bullet\}$ (вставка, array), 176, 197
 \gg (\gg), 63
 \sqcup , 43
 $\@$, 151, 249
 $\@\{\bullet\}$, 76, 183
 $\@.$ (amscd), 125
 $\@(\{\bullet\}\{\bullet\}(\{\bullet\}\{\bullet\}))$ (amscd), 125
 $\@(\{\bullet\}\{\bullet\})\{\bullet\}$ (amscd), 125
 $\@<\{\bullet\}<\{\bullet\}<$ (amscd), 125
 $\@=$ (amscd), 125
 $\@>\{\bullet\}>\{\bullet\}>$ (amscd), 125
 $\@A\{\bullet\}A\{\bullet\}A$ (amscd), 125
 $\@biblabel\{\bullet\}$, 151
 $\@centercr$, 232
 $\@evenfoot\{\bullet\}$, 213
 $\@evenhead\{\bullet\}$, 213
 $\@listi$, 225
 $\@listii$, 223, 225
 $\@listiii$, 225
 $\@listiv$, 225
 $\@listv$, 225
 $\@listvi$, 225
 $\@oddfoot\{\bullet\}$, 213
 $\@oddhead\{\bullet\}$, 213
 $\@openbib@code$, 230
 $\@V\{\bullet\}V\{\bullet\}V$ (amscd), 125
 $\@|$ (amscd), 125
 $\@[, \backslash]$, 89
 $\hat{\cdot}$, 19, 97
 $\tilde{\cdot}$, 19, 41
 $\{$, $\}$, 19
 $_$, 19, 97
 \backslash' (\grave{a}), 60, 292
 $\backslash\prime$ (\grave{a}), 63
 $|$, 74
A
 $\backslashacute\{\bullet\}$ (\acute{a}) (математический
акцент), 308
 $\backslashaddcontentsline\{\bullet\}\{\bullet\}\{\bullet\}$,
203
 $\backslashaddlinespace[]$ (booktabs),
181
 $\backslashaddtocontents\{\bullet\}\{\bullet\}$, 202
 $\backslashaddtocounter\{\bullet\}\{\bullet\}$, 33
 $\backslashaddtolength\{\bullet\}\{\bullet\}$, 39
AdobeReader (программа), 16, 62
 $\backslashadvance\bullet\textit{by}\bullet$ (команда T_EX),
223
 $\backslashafterpage\{\bullet\}$ (afterpage), 204
 $\backslashallowdisplaybreak[]$
(amsmath), 116
 \backslashAMSautorefname (hyperref), 271

`\and` (ifthen), 265
`\appendix`, 30
`\appendixautorefname`
 (hyperref), 271
`\appendixname`, 149
`\arrayrulecolor{•}[•]`
 (colortbl), 196
`\abstractname`, 149
`\author{•}`, 28
`\autoref{•}` (hyperref), 270

B

`b{•}` (колонка, аргумент), 176
`\b{•}` (a), 60, 292
`\bar{•}` (\bar{a}) (математический
 акцент), 308
`\begin{•}[•]{•}`, 20
`\bf`, 57
`\bfseries`, 56
`\bibitem[•]{•}`, 131
`\bibliographystyle{•}`, 134
`\bibliography{•}`, 133
`\bibname`, 149
BibTeX (программа), 15, 132–134
`\Big•`, `\Bigl`, `\Bigm`, `\Bigr`,
 100
`\big•`, `\bigl`, `\bigm`, `\bigr`,
 100
`\Bigg•`, `\Biggl`, `\Biggm`, `\Biggr`,
 100
`\bigg•`, `\biggl`, `\biggm`, `\biggr`,
 100
`\bigskip`, 47
`\bm{•}` (bm), 264
`\boldsymbol{•}` (amsbsy), 93
`\boolean{•}` (ifthen), 265
`\bottomrule[•]` (booktabs), 182
`\bottomrule[•]` (booktabs), 79
BoundingBox, 67–73

`\breve{•}` (\breve{a}) (математический
 акцент), 308

C

`\C{•}` (\ddot{a}), 60, 292
`\c{•}` (\grave{a}), 60, 292
`\caption*{•}` (longtable), 189
`\caption[•]{•}`, 83, 189
`\cellcolor[•]{•}` (colortbl),
 195
`\centering`, 45
`\chapter*{•}`, 30
`\chapterautorefname`
 (hyperref), 271
`\chaptermark{•}`, 214
`\chaptername`, 149
`\chapter[•]{•}`, 30
`\check{•}` (\check{a}) (математический
 акцент), 308
`\cite[•]{•}`, 131
`\cleardoublepage`, 204
`\clearpage`, 204, 268
`\cline{•}`, 75
`\cmidrule•{•}` (booktabs),
 79, 182
`\colorbox[•]{•}{•}` (color),
 174
`\color[•]{•}` (color), 172
`\columncolor[•]{•}[•][•]`
 (colortbl), 194
`\contentsline{•}{•}{•}`, 201
`\contentsline{•}{•}{•}{•}`
 (hyperref), 201
`\contentsname`, 149
`\counterwithin{•}{•}`, 161
`\counterwithout{•}{•}`, 161
cp1251 (кодировка текста), 287
cp1252 (кодировка текста), 287

D

`D{•}{•}{•}` (колонка, `dcolumn`),
178

`\d{•}` (а), 60, 292

`\date{•}`, 28

`\ddddot{•}` (¨¨¨)
(математический
акцент), 308

`\dddot{•}` (¨¨) (математический
акцент), 308

`\ddot{•}` (¨) (математический
акцент), 308

`\DeclareMathOperator*{•}{•}{•}`
(`amsopn`), 96

`\DeclareMathOperator*{•}{•}{•}`
(`amsopn`), 100

`\DeclareTextSymbolDefault{•}{•}`,
288

`\def • # {•}` (команда `TEX`), 150

`\definecolor{•}{•}{•}` (`color`),
171, 172

`\DefineNamedColor{•}{•}{•}{•}`
(`color`), 172

definition (стиль теорем), 122

`\depthof{•}` (`calc`), 169

`\descriptionlabel{•}`, 228

`\dfrac{•}{•}` (`amsmath`), 97

`\displaybreak[•]` (`amsmath`),
116

`\displaystyle`, 103

`\documentclass{•}{•}`, 22

`\dot{•}` (˙) (математический
акцент), 308

`\dotfill`, 45

`\dots` (`amsmath`), 64

`\doublerulesepcolor{•}[•]`
(`colortbl`), 196

`dvips` (программа), 15

E

`\em`, 56

`\emph{•}`, 56

`\endfirsthead` (`longtable`), 188

`\endfoot` (`longtable`), 188

`\endhead` (`longtable`), 188

`\endinput`, 199

`\endlastfoot` (`longtable`), 188

`\end{•}`, 20

`\enlargethispage*{•}`, 50

`\enskip`, 42

`\enspace`, 42

`\ensuremath{•}`, 148

`\epstopdfsetup{•}` (`epstopdf`),
267

`\eqref{•}` (`amsmath`), 32

`\equal{•}{•}` (`ifthen`), 265

`\equationautorefname`
(`hyperref`), 271

`\extracolsep{•}`, 192

F

`\f{•}` (â), 60, 292

`\fbox{•}`, 71, 156

`\fcolorbox[•]{•}{•}{•}`
(`color`), 174

`\figureautorefname` (`hyperref`),
271

`\figurename`, 149

`\firstline` (`array`), 179

`\flushbottom`, 49

`\footnotemark[•]`, 35

`\footnotesize`, 58

`\footnotetext[•]{•}`, 35

`\footnote[•]{•}`, 35

`\frac{•}{•}`, 97

`\framebox[•][•]{•}`, 157

G

`\gdef • # {•}` (команда `TEX`), 150

Ghostscript (программа), 9
\glossaryentry{•}{•}, 256
\glossary{•}, 256
\graphicspath{•} (graphicx),
69
\grave{•} (à) (математический
акцент), 308
GSview (программа), 9

Н

\H{•} (ñ), 60, 292
\hat{•} (â) (математический
акцент), 308
\heightof{•} (calc), 169
\hfil, 43
\hfill, 43, 52, 78
\Hfootnoteautorefname
(hyperref), 271
\hhline{•} (hhline), 184
\hline, 75
\hphantom{•}, 104
\href{•}{•} (hyperref), 270
\hrulefill, 45
\hsize (длина), 194
\hspace*{•}, 42
\hss, 226
\huge, 58
\huge, 58
\hyperpage{•}, 323
\hyperref{•}{•} (hyperref), 270
\hypersetup{•} (hyperref), 275

I

\IeC{•}, 62
\IfFileExists{•}{•}{•}, 266
\ifthenelse{•}{•}{•} (ifthen),
265
\includegraphics*{•}{•}
(graphicx), 69

\includeonly{•}, 199
\include{•}, 198
\indent, 42
\indexentry{•}{•}, 248
\indexname, 149
\indexprologue{•}{•}
(imakeidx), 253
\indexsetup{•} (imakeidx), 254
\indexspace, 261
\index{•}, 248
\index[•]{•} (imakeidx), 253
\input{•}, 198
\intertext{•} (amsmath), 105
iso8859-1 (кодировка текста), 287
\isodd{•} (ifthen), 265
\it, 57
\item[•], 127, 248
\Itemautorefname (hyperref),
271
\itemlabels (нестандартная),
153
\itemlabelset{•}{•}{•}{•}
(нестандартная), 153
\itshape, 56

J

JabRef (программа), 9

К

\k{•} (q), 60, 292
kpsewhich (программа), 285

L

\labelenumiii{•}, 227
\labelenumii{•}, 227
\labelenumiv{•}, 227
\labelenumi{•}, 227
\labelitemfont, 153, 226
\labelitemi{•}, 226
\labelitemii{•}, 226

`\labelitemiii{•}`, 226
`\labelitemiv{•}`, 226
`\label{•}`, 32
`\LARGE`, 58
`\large`, 58
`\lastline` (array), 179
`\left•`, 101
`\leftmark`, 214
`\lengthtest{•}` (ifthen), 265
`\let•=•` (команда `TeX`), 150
`\limits`, 99
`\linebreak[•]`, 41
`\listfigurename`, 149
`\listoffigures`, 201
`\listoftables`, 201
`\listtablename`, 149

M

`m{•}` (колонка, array), 176
`\makeatletter`, 151
`\makeatother`, 151
`\makebox[•][•]{•}`, 156
`\makeglossary` (makeidx), 256
`\makeindex` (makeidx), 248
`MakeIndex` (программа), 15, 248
`\makeindex[•]` (imakeidx), 252
`\makelabel{•}`, 225
`\MakeLowercase{•}`, 215
`\maketitle`, 29
`\MakeUppercase{•}`, 215
`\marginparpush`, 212
`\marginpar[•]{•}`, 210
`\markboth{•}{•}`, 214
`\markright{•}`, 214
`\mathstrut`, 105
`\mbox{•}`, 41
`\mdseries`, 56
`\medskip`, 47
`metafont` (программа), 285

`\midrule[•]` (booktabs), 182
`\midrule[•]` (booktabs), 79
`MikTeX` (программа), 8, 9
`mktexlsr` (программа), 283
`\morecmidrules` (booktabs), 183
`\mspace{•}` (amsmath), 104
`\multicolumn{•}{•}{•}`, 78
`\multirowsetup` (multirow), 81
`\multirow[•]{•}[•]{•}[•]{•}`
 (multirow), 80

N

`\newblock`, 24, 230
`\newboolean{•}` (ifthen), 266
`\newcolumntype{•}[•]{•}`
 (array), 179
`\newcommand*{•}[•][•]{•}`,
 147
`\newcounter{•}[•]`, 160
`\newenvironment*`
`{•}[•][•]{•}{•}`, 152
`\newlength{•}`, 164
`\newpage`, 48, 52, 190
`\newsavebox{•}`, 159
`\newtheorem*{•}{•}` (amsthm),
 234
`\newtheorem*{•}{•}` (amsthm),
 119
`\newtheorem{•}{•}[•]`, 121
`\newtheorem{•}[•]{•}`, 120
`\nocite{•}`, 135
`\nofiles`, 203
`\noindent`, 42
`\nolimits`, 99
`\nolinkurl{•}` (hyperref), 272
`\nonumber`, 111
`\nopagebreak[•]`, 48
`\nopagecolor` (color), 174
`\normalfont`, 57

`\normalsize`, 58
`\not` (ifthen), 265
`\notag` (amsmath), 111
`\numberline{•}`, 202

O

`\operatorname*{•}{•}`
(amsopn), 96
`\operatorname*{•}{•}`
(amsopn), 100
`\or` (ifthen), 265
OT2 (кодировка шрифта), 288

P

`p{•}` (колонка), 76
`\p@enumii`, 227
`\p@enumiii`, 227
`\p@enumiv`, 227, 230
`\pageautorefname` (hyperref),
271
`\pagebreak`, 190
`\pagebreak[•]`, 48
`\pagecolor[•]{•}` (color), 174
`\pagenumbering{•}`, 216
`\pageref*{•}` (hyperref), 270
`\pageref{•}`, 32
`\pagestyle{•}`, 212
`\par`, 224
`\paragraph*{•}`, 30
`\paragraphautorefname`
(hyperref), 271
`\paragraph[•]{•}`, 30
`\parbox[•]{•}[•]{•}{•}`, 53
`\part*{•}`, 30
`\partautorefname` (hyperref),
271
`\partname`, 149
`\part[•]{•}`, 30
pdf \LaTeX (программа), 13

pdf \TeX (программа), 13
`\phantomsection` (hyperref), 273
``, 104
plain (стиль теорем), 122
`\pmb{•}` (amssys), 276
`\printindex`, 248
`\printindex[•]` (imakeidx), 253
`\protect`, 34, 202
`\provideboolean{•}` (ifthen),
266
`\providecommand*{•}[•][•]{•}`,
147

Q

`\qqquad`, 42
`\quad`, 42

R

`\r{•}` (ã), 60, 292
`\raggedbottom`, 49
`\raggedleft`, 45
`\raggedright`, 45
`\raisebox{•}[•][•]{•}`, 157
`\ratio{•}{•}` (calc), 168
`\real{•}` (calc), 164
`\ref*{•}` (hyperref), 270
`\reflectbox{•}` (graphicx), 85,
158
`\refname`, 149
`\refstepcounter`, 272
`\refstepcounter{•}`, 161
`\ref{•}`, 32
`\relax`, 261
remark (стиль теорем), 122
`\renewcommand*{•}[•][•]{•}`,
147
`\renewenvironment*`
`{•}[•][•]{•}{•}`, 152
`\resizebox*{•}{•}{•}`
(graphics), 158

`\right`, 101
`\rightmark`, 214
`\rm`, 57
`\rmfamily`, 56
`\rotatebox[•]{•}{•}`
 (graphicx), 158
`\rowcolor[•]{•}[•][•]`
 (colortbl), 195
`\rule[•]{•}{•}`, 19

S

`\savebox{•}[•][•]{•}`, 159
`\sbox{•}{•}`, 159
`\sc`, 57
`\scalebox{•}[•]{•}` (graphicx),
 157
`\scriptscriptstyle`, 103
`\scriptsize`, 58
`\scriptstyle`, 103
`\scshape`, 56
`\section*{•}`, 30
`\sectionautorefname`
 (hyperref), 271
`\sectionmark{•}`, 214
`\section[•]{•}`, 30
`\seealsoname`, 250
`\seename`, 250
`\setboolean{•}{•}` (ifthen), 266
`\setcounter{•}{•}`, 33
`\setlength{•}{•}`, 39
`\settodepth{•}{•}`, 169
`\settoheight{•}{•}`, 169
`\settototalheight{•}{•}`
 (calc), 169
`\settowidth{•}{•}`, 169
`\sf`, 57
`\sffamily`, 56
`\sideset{•}{•}{•}` (amsmath),
 100

`\sl`, 57
`\slshape`, 56
`\small`, 58
`\smallskip`, 47
`\specialrule{•}{•}{•}`
 (booktabs), 182
splitindex (программа), 251
`\splitindexoptions{•}`
 (imakeidx), 255
`\sqrt{•}{•}`, 96
`\stepcounter{•}`, 161
`\stretch{•}`, 165
`\subitem`, 248
`\subparagraph*{•}`, 30
`\subparagraph[•]{•}`, 30
`\subsection*{•}`, 30
`\subsectionautorefname`
 (hyperref), 271
`\subsectionmark{•}`, 215
`\subsection[•]{•}`, 30
`\substack{•}` (amsmath), 98
`\subsubitem`, 248
`\subsubsection*{•}`, 30
`\subsubsectionautorefname`
 (hyperref), 271
`\subsubsection[•]{•}`, 30
`\swapnumbers` (amsthm), 121

T

`\t{•}` (aa), 60, 292
T1 (кодировка шрифта), 62, 288,
 290
T2A (кодировка шрифта), 62,
 288, 290
T2B (кодировка шрифта), 288
T2C (кодировка шрифта), 288
`\tableautorefname` (hyperref),
 271
`\tablename`, 149

`\tableofcontents`, 31
`\tabularnewline[\bullet]`, 191
`\tag*{ \bullet }` (amsmath), 112
TeXindy (программа), 247, 321
TeXstudio (программа), 10–12, 280
`\textasciicircum` (\wedge), 19
`\textasciitilde` (\sim), 19
`\textbackslash` (\backslash), 19
`\textbf{ \bullet }`, 56
`\textcircled{ \bullet }` \odot , 60
`\textcolor[\bullet]{ \bullet }{ \bullet }` (color), 172
`\textcommabelow{ \bullet }` (\a), 60, 292
`\textemdash` (---), 63
`\textendash` (--), 63
`\textit{ \bullet }`, 56
`\textmd{ \bullet }`, 56
`\textrm{ \bullet }`, 56
`\textsc{ \bullet }`, 56
`\textsf{ \bullet }`, 56
`\textsl{ \bullet }`, 56
`\textstyle`, 103
`\texttt{ \bullet }`, 56
`\textup{ \bullet }`, 56
`\text{ \bullet }` (amstext), 105
`\tfrac{ \bullet }{ \bullet }` (amsmath), 97
`\theenumiii{ \bullet }`, 227
`\theenumii{ \bullet }`, 227
`\theenumiv{ \bullet }`, 230
`\theenumiv{ \bullet }`, 227
`\theenumi{ \bullet }`, 227
`\theoremautorefname` (hyperref), 271
`\theoremstyle{ \bullet }` (amsthm), 122
`\thedлина`, 39
`\thesчетчик`, 34, 163
`\thispagestyle{ \bullet }`, 52, 212
`\tilde{ \bullet }` (\tilde{a}) (математический акцент), 308
`\tiny`, 58
`\title{ \bullet }`, 28
`\today`, 28
`\toprule[\bullet]` (booktabs), 182
`\toprule[\bullet]` (booktabs), 79
`\totalheightof{ \bullet }` (calc), 169
TS1 (кодировка шрифта), 288, 290
`\tt`, 57
`\ttfamily`, 56
U
`\U{ \bullet }` (\AA), 60, 292
`\u{ \bullet }` (\aa), 60, 292
`\underline{ \bullet }`, 45, 218
`\upshape`, 56
`\url{ \bullet }` (hyperref), 272
`\usebox{ \bullet }`, 160
`\usecounter{ \bullet }`, 225
`\UseMicrotypeSet[\bullet]{ \bullet }` (microtype), 269
`\usepackage[\bullet]{ \bullet }`, 25
`\UseTextSymbol{ \bullet }{ \bullet }`, 288
utf8 (кодировка текста), 59, 290
V
`\v{ \bullet }` (\v{a}), 60, 292
`\value{ \bullet }`, 163
`\vec{ \bullet }` (\vec{a}) (математический акцент), 308
`\verb* \bullet \dots \bullet` , 64
`\vfil`, 48
`\vfill`, 48
`\vline width \bullet` , 184
`\vphantom{ \bullet }`, 104
`\vspace*{ \bullet }`, 47

W

`W•{•}` (колонка, аргумент), 176
`w•{•}` (колонка, аргумент), 176
`\whiledo{•}{•}` (ifthen), 265
`\widehat{•}` (\widehat{abc})
(математический акцент), 308
`\widetilde{•}` (\widetilde{abc})
(математический акцент), 308
`\widthof{•}` (calc), 169

X

X2 (кодировка шрифта), 288
xindy (программа), 15, 247
`\xspace` (xspace), 264

Б

библиография, 130
BibTeX, 132
записи, 141
компиляция, 132
поля записей, 141
программы, 136
стили, 134
добавление источников, 135
источники, 131
пакеты
biblatex, 245
chapterbib, 200
cite, 237
gost, 135, 240
natbib, 240
подключение баз, 133
ссылки, 131, 243
стили цитирования, 242
боксы, 36
министраницы, 52, 53

однострочные, 41
операции
вращение, 158
использование, 160
масштабирование, 157, 158
отражение, 158
сдвиг, 157
сохранение, 159
рамка, 71

буквы

диакритические знаки, 60, 292
кириллица, 61, 296–299
латиница, 61, 293–295

Г

гlossарий, 256
вывод, 257
инициация, 256
передача элемента, 256
размер шрифта, 260
создание элемента, 256
группы, 20

Д

декларации, 21
диаграммы
коммутативные, 123
длины, 37
вычисления, 166
единицы, 38, 104
изменение, 39
печать, 39
создание, 164
упругие, 165

З

заметки на полях, 208

К

кавычки, 63
кегель, 38
классы документов, 23
 настройки, 23
колонтитулы, 212
команды, 17
 аргументы, 18
 перемещаемые, 34
 декларации, 21
 изменение, 147
 параметры, 18
 создание, 147
 хрупкие, 34
комментарии, 17
компиляция, 13
 библиографии, 15, 132–135
 документа, 13, 280–283
 указателя, 15, 321–324

Л

лигатура, 63

М

математика
 блоки, 106, 107
 векторы, 92, 93
 дроби, 97
 индексы, 97
 пределы, 98, 99, 277
 корни, 96
 матрицы, 108
 нумерация формул, 110,
 277
 связанная, 111
операторы
 бинарные, 94, 313
 большие, 99, 277, 310
пробелы, 104

разделители, 100, 311
размер символов, 102
символы, 94, 312
 акценты, 92, 308
 алфавиты, 90, 92
 греческие буквы, 92, 309
соотношения, 94, 314, 315
 с отрицанием, 317
 стрелки, 95, 315, 316
текст, 105
точки и многоточия, 95,
318
фантомы, 104
формулы
 в строке, 88
 вынесенные, 88
 нумерованные, 110
функции, 95, 277, 319
 пределы, 99, 320
 создание, 96

метки, 32

О

оглавление, 31
окружения, 20
отбивка, 45
отступы, 42

П

пакеты, 26
плавающие объекты, 82
предметный указатель, 246, 248
 вывод, 248
 гlossарий, 256
 инициация, 248
 передача элемента, 248
 размер шрифта, 260
 создание элемента, 248
 изменение шрифта, 249

- ключ сортировки, 249
- комментарий, 250
- структура , 248
- шрифт страницы, 249
- приложения, 31
- примечания, 35
- пробелы
 - в формулах, 104
 - вертикальные, 41, 47
 - горизонтальные, 42
 - неразрывный (~), 41
 - фантомы, 104
- программы
 - AdobeReader, 16, 62
 - biber, 135
 - BibTeX, 15, 132–134
 - bibtex8, 135
 - dvips, 15
 - epstopdf, 68
 - Ghostscript, 9
 - GSview, 9
 - JabRef, 9, 136
 - kpsewhich, 285
 - MakeIndex, 15, 248
 - metafont, 285
 - MikTeX, 8, 9
 - mktexlsr, 283
 - pdfL^AT_EX, 13
 - pdfTeX, 13
 - ps2eps, 69
 - splitindex, 251
 - texindy, 247, 321
 - TeXstudio, 10–12, 280
 - xindy, 15, 247
- пружины
 - вертикальные, 48
 - горизонтальные, 43

Р

- разделы документа, 30
- заголовки, 149, 271
- рисунки
 - вращение, 70
 - вставка иллюстраций, 69
 - масштабирование, 70
 - обрезание полей, 71
 - подписи, 83

С

- символы, 60, 288, 290–292
 - неопределенные, 62
 - служебные, 19
- списки, 126, 220
 - библиография, 130, 229
 - вложенность, 127, 129
 - гlossарий, 256
 - метки, 127, 226, 227
 - настраиваемые, 220
 - нечисловые, 127, 226
 - числовые, 128, 228
 - оглавление, 31, 201
 - описания, 128, 228
 - предметный указатель, 246
 - рисунков, 201
 - таблиц, 201
- ссылки
 - гиперссылки, 270
 - на источник, 131, 243
 - на объект, 32
 - на страницу, 32
 - на формулу, 32
- стихи, 231
- строки
 - отступ, 42
 - перенос, 40
 - разрыв
 - с выравниванием, 41

- без выравнивания, 41, 47
- структура, 40
- счетчики, 33
 - вычисления, 163
 - значения, 34, 163
 - изменение, 33
 - обновление, 161
 - связывание, 161
 - создание, 160
 - установка, 33
 - форматы вывода, 162

Т

- таблицы, 73
 - вид
 - длинные, 186
 - вставки, 176
 - выравнивание ячеек, 77
 - колонки, 76, 176, 178, 194
 - настройки, 180
 - объединение
 - колонок, 78
 - строк, 79
 - отчеркивание
 - колонок, 74
 - строк, 75, 79
 - ячеек, 75
 - подписи, 83
 - привязка к строке, 75
 - разделители, 76, 176
 - фон
 - колонок, 194
 - строки, 195
 - ячейки, 195
 - цвет линий, 196
- текст
 - выделение, 46
 - выравнивание, 45
 - кодировки, 59, 287

- подчеркивание, 45
- пробелы, 16
- разбиение
 - без выравнивания, 48
 - с выравниванием, 48
- рифты, 53
- теоремы
 - вывод номера, 121
 - доказательства, 120
 - нумерация
 - без номера, 119
 - подчиненная, 121
 - с номером, 120
 - стиль оформления, 122
- тире, 63

Ц

- цветная верстка, 170
 - набор цветов, 171, 173
 - палитры, 171
 - синтез цвета, 171
- таблицы
 - фон колонок, 194
 - фон строки, 195
 - фон ячейки, 195
 - цвет линий, 196
- фон, 174
- фон страницы, 174
- цвет текста, 172

Ш

- рифты, 53
 - кодировки, 62, 288
 - контрастность, 54
 - насыщенность, 54
 - начертание, 54
 - основной шрифт, 57
 - параметры гарнитур, 56
 - размеры, 58
 - часто используемые, 57

Окружения

Окружения с именами `name` и `name*` и одинаковым синтаксисом представлены общим именем `name*`. Символы `{•}` и `[•]` указывают на наличие аргументов и параметров.

A

`abstract` (аннотация), 29
`alignat* {•}` (формула), 115
`aligned` (блок), 110
`align*` (формула), 114
`array {•}` (блок), 109
`array [•]{•}` (блок), 73

B

`bfseries` (**жирный шрифт**), 56
`Bmatrix` (матрица), 108
`bmatrix` (матрица), 108

C

`cases` (блок), 109
`CD` (блок, `amscd`), 123
`center` (центрирование), 45, 233
`comment` (`verbatim`), 264

D

`description` (список), 128, 228
`document` (документ), 26

E

`enumerate` (список), 128, 228
`enumerate [•]` (`enumerate`), 264
`eqnarray*` (формула), 114
`equation*` (формула), 112

F

`figure* [•]` (плавающий объект),
82
`flalign*` (формула), 115
`flushleft` (выравнивание слева), 45,
233
`flushright` (выравнивание справа),
45, 233
`footnotesize` (мелкий шрифт), 58

G

`gathered` (блок), 107
`gather*` (формула), 113

H

`Huge` (крупный шрифт), 58
`huge` (крупный шрифт), 58

I

itemize (список), 127, 226
itshape (*курсив*), 56

L

landscape (ориентация страницы),
268
LARGE (крупный шрифт), 58
Large (крупный шрифт), 58
large (крупный шрифт), 58
list {•}{•} (список), 220
longtable [•]{•} (longtable), 186
lrtext {•} (бок), 159

M

matrix (матрица), 108
mdseries (стандартная
контрастность), 56
minipage [•][•][•]{•} (бок),
52
multicols {•}[•][•] (multicol),
264
multiline* (формула), 113
mytemize [•] (нестандартное),
153

N

normalsize (стандартный размер
шрифта), 58

P

pmatrix (матрица), 108
proof [•] (доказательство), 120

Q

quotation (отбивка текста), 45,
231
quote (отбивка текста), 45, 231

R

rmfamily (шрифт Roman), 56

S

scriptsize (мелкий шрифт), 58
scshape (КАПИТЕЛЬ), 56
sffamily (шрифт Serif), 56
slshape (*наклонный шрифт*), 56
small (мелкий шрифт), 58
smallmatrix (матрица), 108
split (блок), 109, 277
subarray {•} (математика), 98
subequations (математика), 111

T

tabbing (таблица), 73
table* [•] (плавающий объект),
82
tabular* {•}[•]{•} (таблица),
192
tabularx {•}[•]{•} (tabularx),
193
tabular [•]{•} (таблица), 75
thebibliography {•}
(библиография), 130,
229
theindex (гlossарий), 257
theindex (указатель), 248
tiny (крошечный шрифт), 58
trivlist (список), 233
ttfamily (шрифт Typewriter), 56

U

upshape (прямой шрифт), 56

V

verbatim* (код), 65, 233
verse (стихи), 232
Vmatrix (матрица), 108
vmatrix (матрица), 108

X

xalignat* {•} (формула), 116

Длины, счетчики, КОНСТАНТЫ

Для констант, статических счетчиков и длин, не зависящих от размера шрифта или параметров страницы, в скобках приведены значения, присваиваемые стандартными классами. Для динамических счетчиков указаны ассоциированные объекты.

В указатель также внесены команды, управляющие счетчиками и длинами, аргументы которых обозначены символами {•}.

`\@listdepth` (list), 225

A

`\addtocounter{•}{•}`, 33

`\addtolength{•}{•}`, 39

`\Alph{•}` (формат A–Z), 162

`\alph{•}` (формат a–z), 162

`\arabic{•}` (формат 0–9), 162

`\arraycolsep` (5pt), 180

`\arrayrulewidth` (0.4pt), 180

`\arraystretch` (1), 180

`\Asbuk{•}` (формат A–Я), 162

`\asbuk{•}` (формат a–я), 162

B

`\baselineskip`, 38, 47

`\baselinestretch` (1.0), 47

`\bibindent` (1.5em), 222

`\bigskipamount` (12pt±4pt), 47

`\bottomfraction` (0.3), 205

`bottomnumber` (1), 205

C

`chapter` (главы), 33

`cm` (0.396in), 38

`\cmidrulesep` (`\doublerulesep`,
booktabs), 183

`\columnsep` (10pt), 38, 208

`\columnseprule` (0pt), 208

`\counterwithin{•}{•}`, 161

`\counterwithout{•}{•}`, 161

D

`\dblfloatpagefraction` (0.5),
205

`\dblfloatsep`, 205

`\dbltextfloatsep`
(20pt+2pt-4pt), 205
`\dbltopfraction` (0.7), 205
`dbltopnumber` (2), 205
`\depth` (боксы), 156
`\doublerulesep` (2pt), 180

E

`em` (▷M◁), 38
`enumi` (списки), 33
`enumii` (списки 2 уровня), 33
`enumiii` (списки 3 уровня), 33
`enumiv` (списки 4 уровня), 33
`equation` (формулы), 33
`\evensidemargin`, 209
`ex` ($\frac{x}{x}$), 38
`\extrarowheight` (0pt, array),
180

F

`\fboxrule` (0.4pt), 157
`\fboxsep` (3pt), 71, 157
`figure` (рисунки), 33
`fil` (единица жесткости), 165
`\fill` (0pt+1fill), 38, 165
`fill` (единица жесткости), 165
`filll` (единица жесткости), 165
`\floatpagefraction` (0.5), 205
`\floatsep` (12pt±2pt), 205
`\fnsymbol{•}` (формат
*,†,‡,§,¶,||,**,††,‡‡), 162
`footnote` (сноски), 33
`\footskip`, 209

H

`\headheight`, 209
`\headsep`, 209
`\height` (боксы), 156
`\hoffset` (0pt), 38, 209

I

`in` (25.4мм), 38
`\intextsep`, 205
`\itemindent`, 224
`\itemsep`, 222, 223

L

`\labelsep` (0.5em), 222, 224
`\labelwidth`, 224
`\leftmargin`, 224
`\leftmargini`, 222, 224
`\leftmarginii` (2.2em), 222, 224
`\leftmarginiii` (1.87em), 222,
224
`\leftmarginiv` (1.7em), 222, 224
`\leftmarginv`, 222, 224
`\leftmarginvi`, 222, 224
`\linewidth`, 38
`\listparindent`, 224
`\LTcapwidth` (4in, longtable), 190
`LTchunksize` (20, longtable), 190
`\LTleft` (`\fill`, longtable), 190
`\LTpost` (`\bigskipamount`,
longtable), 190, 206
`\LTpre` (`\bigskipamount`,
longtable), 190, 206
`\LTRight` (`\fill`, longtable), 190

M

`\marginparpush`, 209
`\marginparsep`, 209
`\marginparwidth`, 209
`MaxMatrixCols` (10, amsmath), 107
`\medskipamount` (6pt±2pt), 47
`mm` (2.845pt), 38
`mpfootnote` (сноски
министраниц), 33
`mu` (1/18 em), 104

N
`\newlength{•}`, 164

O
`\oddsidemargin`, 209

P
page (страницы), 33
`\paperheight`, 38, 209
`\paperwidth`, 38, 209
paragraph (разделы 4 уровня), 33
parentequation (система уравнений), 33
`\parindent`, 38
`\parsep`, 222, 223
`\parskip (0pt+1pt)`, 38, 223
part (части документа), 33
`\partopsep`, 222, 224
pt (0.351мм), 38

R
`\ratio{•}{•}` (calc), 168
`\refstepcounter{•}`, 161
`\rightmargin`, 224
`\roman{•}` (формат c,i,l,m,v,x), 162
`\Roman{•}` (формат C,I,L,M,V,X), 162

S
secnumdepth (2, 3), 162
section (разделы), 33
`\setcounter{•}{•}`, 33
`\setlength{•}{•}`, 39
`\smallskipamount (3pt±1pt)`, 47
`\stepcounter{•}`, 161
`\stretch{•}`, 165
subparagraph (разделы 5 уровня), 33
subsection (разделы 2 уровня), 33

subsubsection (разделы 3 уровня), 33

T
`\tabcolsep (6pt)`, 180
table (таблицы), 33
`\textfloatsep (20pt+2pt-4pt)`, 205
`\textfraction (0.2)`, 205
`\textheight`, 38, 209
`\textwidth`, 38, 209
`\thedлина`, 39
`\thesчетчик`, 34, 163
tocdepth (2, 3), 162
`\topfraction (0.7)`, 205
`\topmargin`, 209
topnumber (2), 205
`\topsep`, 222, 223
`\totalheight (боксы)`, 156
totalnumber (3), 205

U
`\usecounter{•}`, 225

V
`\value{•}`, 163
`\voffset (0pt)`, 38, 209

W
`\width (боксы)`, 156

Д
длины, 37
вычисления, 166
единицы, 38, 104
изменение, 39
печать, 39
создание, 164
упругие, 165

С

счетчики, 33

вычисления, 163

значения, 34, 163

изменение, 33

обновление, 161

связывание, 161

создание, 160

установка, 33

форматы вывода, 162

Пакеты

Символы [•] указывают на наличие настроек.

A

afterpage [•] (tools), 204
amscd (amslatex), 123
amsmath (amslatex), 88, 306
amslatex (коллекция), 88, 119
amsmath (amslatex), 88
amssymb (amslatex), 88, 306
amsthm (amslatex), 119
array (tools), 176
article [•] (класс), 23

B

babel [•], 40
base (коллекция), 265
biblatex [•], 245
bm (tools), 264
booktabs, 79, 181
book [•] (класс), 23

C

calc (tools), 163, 166
caption [•], 234
chapterbib [•] (библиография),
200, 242
cite [•] (библиография), 131

cmr, 62, 290
cm-super, 9
colortbl, 194
colortbl [•], 166
color [•] (latex-graphics), 170

D

dcolumn (tools), 178
delarray (tools), 264

E

elsarticle [•] (класс), 29, 132
enumerate (tools), 264
epstopdf-pkg (коллекция), 267
epstopdf [•] (epstopdf-pkg), 267
eucal [•] (amslatex), 91, 306
extarticle [•] (класс, extsizes), 58
extbook [•] (класс, extsizes), 58
extletter [•] (класс, extsizes), 58
extproc [•] (класс, extsizes), 58
extreport [•] (класс, extsizes), 58,
263
extsizes (коллекция), 58

F

fancyhdr [•], 219

fontenc [●], 288
fontspec [●] (X_YL^AT_EX, Lua^AT_EX),
279
ftnright (tools), 264

G

geometry [●], 27
glossaries [●], 261
gost (bibtex), 135, 240
graphicx [●] (latex-graphics), 69

H

hhline (tools), 184, 197
hyperref [●], 33, 132, 269

I

ifthen (tools), 265
imakeidx [●], 251
indentfirst, 42
inputenc [●], 59

L

latex-graphics (коллекция), 69,
170, 268
longtable (tools), 186
lscapex [●] (latex-graphics), 268

M

makeidx, 248
microtype [●], 268
multicol (tools), 264
multirow [●], 79

N

natbib [●] (библиография), 240

P

pdfscape [●], 268
proc [●] (класс), 23

R

report [●] (класс), 23
revtex4-2 [●] (класс), 29
revtex [●] (класс), 132

T

tabularx (tools), 193
textcomp, 60, 290
theorem (tools), 122
tools (коллекция), 263

U

unicode-math [●] (X_YL^AT_EX,
Lua^AT_EX), 279
upgreek [●] (was), 235

V

varioref [●] (tools), 264
verbatim (tools), 264

W

was (коллекция), 235

X

xspace (tools), 264
XY-pic (коллекция), 125
xy [●] (XY-pic), 125

Средства верстки библиографии

В указателе представлены средства BibTeX, программы, пакеты и команды, предназначенные для верстки библиографии и цитирования источников.

Аргументы и параметры команд обозначены символами [•] и {•}. Символы [•] указывают также на наличие настроек пакетов.

В

biber (программа), 135
\bibitem[•]{•}, 131
\bibliography{•}, 133
\bibliographystyle{•}, 134
BibTeX (программа), 132–134
bibtex8 (программа), 135

С

\cite[•]{•}, 131
\cite[•][•]{•} (natbib), 244
\Citealp*[•][•]{•} (natbib),
244
\citealp*[•][•]{•} (natbib),
244
\Citealt*[•][•]{•} (natbib),
244
\citealt*[•][•]{•} (natbib),
244

\citedash (cite), 239
\citeform{•} (cite), 239
\citeleft (cite), 238
\citemid (cite), 239
\CiteMoveChars (cite), 239
\citen{•} (cite), 239
\citenum{•} (cite), 239
\citenum{•} (natbib), 245
\citepunct (cite), 239
\Citep*[•][•]{•} (natbib), 244
\citep*[•][•]{•} (natbib), 244
\citeright (cite), 238
\citestyle{•} (natbib), 242
\Citet*[•][•]{•} (natbib), 243
\citet*[•][•]{•} (natbib), 243

Е

epstopdf (программа), 68

Ж
JabRef (программа), 136

Н
\nocite{•}, 135

О
\OverciteFont (cite), 239

Т
thebibliography{•} (окружение),
130

В
вывод баз, 135

З
записи, 141
@article, 137
@book, 137
@booklet, 137
@conference, 137
@inbook, 137
@incollection, 137
@inproceedings, 137
@manual, 137
@masterthesis, 137
@misc, 138
@phdthesis, 137
@proceedings, 137
@string, 142
@techreport, 138
@unpublished, 138

К
компиляция, 132–134

Н
настройка полей
author, 140

journal, 142
title, 142
type, 143

П
пакеты
biblatex[•], 245
chapterbib[•], 242
cite[•], 131
elsarticle [•], 132
gost, 135, 240
natbib [•], 240
revtex [•], 132
подключение баз, 133
поля записей, 141
address, 139
author, 139
booktitle, 139
chapter, 139
crossref, 143
doi, 139
editor, 139
eid, 139
eprint, 139
howpublished, 139
institution, 139
journal, 139
key, 139
language, 139
month, 140
note, 140
number, 140
organization, 140
pages, 140
school, 140
series, 140
title, 140
type, 140
url, 140

volume, 140
year, 140

С

стили, 134
abbr, 134
abbrvnat (natbib), 240
alpha, 134
gost2008, 135
gost2008n (gost, natbib), 240
gost2008ns (gost, natbib), 240
plain, 134

plainnat (natbib), 240
ugost2008n (gost, natbib),
240
ugost2008ns (gost, natbib),
240
unsrt, 134
unsrtnat (natbib), 240
стили цитирования, 242

Я

ярлык записи, 138

Список иллюстраций

1.1.	Рабочее окно программы TeXstudio	11
1.2.	Блок-схема процесса компиляции	14
2.1.	Структура строки текста	40
3.1.	Обработка иллюстраций с некорректными границами	72
3.2.	Стандартная компоновка рисунка	82
3.3.	Компоновка рисунка и подписи, расположенной слева	84
3.4.	Компоновка рисунка, состоящего из двух картинок и подписи, расположенной справа от них	85
5.1.	Создание библиографии с использованием BibTeX	132
7.1.	Загружаемые цвета пакета color	173
9.1	Рисунок, сверстаный вручную	206
9.2.	Макет полосы набора четных и нечетных страниц	209
9.3.	Примеры заметок на полях	211
9.4	Пасутся тигры на полях	211
9.5.	Формат стандартных списков	221
11.1.	Создание предметного указателя с помощью xindy	247
11.2.	Пример глоссария	259

Список таблиц

1.1. Служебные символы	19
1.2. Команды разбивки	30
1.3. Список основных счетчиков	33
2.1. Основные единицы длин	38
2.2. Основные длины	38
2.3. Часто используемые пробелы	42
2.4. Средства выравнивания массива текста	45
2.5. Параметры гарнитур Computer Modern	56
2.6. Стандартные размеры шрифтов	58
2.7. Текстовые символы кодировки utf8	60
2.8. Диакритические знаки	60
2.9. Буквы алфавитов латиницы	61
2.10. Буквы алфавитов кириллицы	61
4.1. Команды и символы математических алфавитов	90
4.2. Греческие буквы	92
4.3. Математические акценты	92
4.4. Различные символы	94
4.5. Бинарные операторы	94
4.6. Соотношения	94
4.7. Стрелки (соотношения)	95
4.8. Точки и многоточия	95
4.9. Математические функции	96
4.10. Большие операторы и функции, имеющие пределы	99

4.11. Разделители	100
4.12. Математические блоки	107
4.13. Математические окружения	110
4.14. Морфизмы коммутативных диаграмм	125
5.1. Стандартные типы и поля библиографических записей	141
6.1. Стандартные заголовки	149
6.2. Форматы вывода счетчиков	162
8.1. Дополнения к синтаксису окружения <code>tabular</code> . . .	176
8.2. Пример организации длинной таблицы	187
9.1. Параметры, регулирующие вывод плавающих объектов	205
9.2. Параметры настройки стандартных списков: длины вертикальных отбивок	222
9.3. Параметры настройки стандартных списков: длины горизонтальных отбивок	222
10.1. Стили оформления ссылок пакета <code>natbib</code>	242
12.1. Контекстные вставки команды <code>\autoref</code>	271
Б.1 Таблица кодировок T1 и T2A	289
Б.2. Текстовые символы	290
Б.3. Команды расстановки диакритических знаков . .	292
Б.4. Буквы алфавитов латиницы	293
Б.5. Буквы алфавитов кириллицы	296
В.1. Математический курсив	301
В.2. Прямой шрифт Roman	302
В.3. Шрифт Sans-Serif	304
В.4. Шрифт Typewriter	305
В.5. Каллиграфический шрифт	305

V.6. Рукописный шрифт	306
V.7. Готический шрифт Fraktur	307
V.8. Ажурный шрифт Double-struck (Blackboard bold)	308
V.9. Математические акценты	308
V.10. Греческие буквы	309
V.11. Большие операторы	310
V.12. Разделители	311
V.13. Буквенные символы	312
V.14. Различные символы	312
V.15. Бинарные операторы	313
V.16. Соотношения	314
V.17. Стрелки (соотношения)	315
V.18. Соотношения с отрицанием	317
V.19. Точки и многоточия	318
V.20. Математические функции	319
V.21. Функции, имеющие пределы	320

Оглавление

Предисловие	3
Часть I. Верстка в \LaTeX	6
Глава 1. Создание документа	7
1.1. Установка и конфигурирование программ	8
1.2. Компиляция и просмотр документа	13
1.3. Основные понятия и синтаксис \LaTeX	16
1.3.1. Команды и окружения	17
1.4. Структура документа	22
1.4.1. Преамбула. Класс документа и пакеты	22
1.4.2. Параметры страницы	27
1.4.3. Титульная часть	28
1.4.4. Разделы	29
1.4.5. Оглавление	31
1.4.6. Метки, ссылки, счетчики	32
1.4.7. Примечания	35
Глава 2. Верстка текста	36
2.1. Длины	37
2.2. Верстка абзацев	39
2.2.1. Выравнивание строк	40

2.2.2.	Отступы	42
2.2.3.	Пробелы	42
2.2.4.	Пружины	43
2.3.	Выделение и выравнивание текста	45
2.4.	Вертикальное выравнивание	47
2.5.	Министранные	52
2.6.	Текстовые шрифты	53
2.6.1.	Гарнитуры Computer Modern	54
2.6.2.	Размеры шрифтов	58
2.7.	Кодировка текста	59
2.8.	Тире, кавычки, точки, запятые	63
2.9.	Вывод текста программы	64
Глава 3.	Рисунки и таблицы	66
3.1.	Иллюстрации	66
3.1.1.	Форматы графики	67
3.1.2.	Обработка иллюстраций	69
3.2.	Верстка таблиц	73
3.2.1.	Выравнивание и разделение колонок	75
3.2.2.	Выравнивание ячеек	77
3.2.3.	Объединение колонок	78
3.2.4.	Объединение строк	79
3.3.	Плавающие объекты	81
Глава 4.	Верстка формул	87
4.1.	Основные элементы и конструкции	89
4.1.1.	Буквы, символы и векторы	89
4.1.2.	Функции и основные конструкции	95
4.1.3.	Большие операторы и пределы	98
4.1.4.	Разделители	100
4.1.5.	Размер символов в формулах	102
4.1.6.	Пробелы в формулах	104
4.1.7.	Текст в формулах	105
4.2.	Математические блоки	106
4.3.	Нумерованные формулы	110

4.3.1.	Нумерация формул	110
4.3.2.	Структура формул	112
4.4.	Верстка сложных формул	117
4.5.	Теоремы	119
4.6.	Коммутативные диаграммы	123
Глава 5.	Библиография и списки	126
5.1.	Списки в тексте	126
5.2.	Список литературы	129
5.2.1.	Библиография и цитирование	130
5.2.2.	Верстка библиографии с помощью BibTeX	132
5.2.3.	Синтаксис библиографических баз	136
Часть II.	Дополнительные возможности	145
Глава 6.	Стандартные операции	146
6.1.	Создание и настройка команд	146
6.2.	Новые окружения	152
6.3.	Операции с боксами	155
6.4.	Операции со счетчиками	160
6.5.	Операции с длинами	164
Глава 7.	Цветная верстка	170
Глава 8.	Таблицы	175
8.1.	Расширенный синтаксис окружения tabular	176
8.2.	Стандартные настройки	180
8.3.	Линии и рамки	180
8.3.1.	Положение линий	180
8.3.2.	Толщина линий	183
8.3.3.	Пересечение линий	184
8.4.	Длинные таблицы	186
8.5.	Таблицы фиксированной ширины	191
8.6.	Цвет в таблицах	194

Глава 9. Настройка документа	198
9.1. Большие документы	198
9.2. Библиография к разделам	200
9.3. Редактирование специальных списков	201
9.4. Управление плавающими объектами	203
9.5. Заметки на полях	208
9.6. Настройка колонтитулов	212
9.6.1. Стилль <code>empty</code>	216
9.6.2. Стилль <code>plain</code>	216
9.6.3. Стилль <code>headings</code>	217
9.6.4. Стилль <code>myheadings</code>	219
9.7. Настройка списков	220
9.7.1. Окружение <code>itemize</code>	226
9.7.2. Окружение <code>enumerate</code>	227
9.7.3. Окружение <code>description</code>	228
9.7.4. Окружение <code>thebibliography</code>	229
9.7.5. Окружения <code>quote</code> , <code>quotation</code> , <code>verse</code>	231
9.7.6. Окружение <code>trivlist</code> и его клоны	233
9.8. \LaTeX по-русски	234
 Глава 10. Стили цитирования	 237
10.1. Оформление ссылок	237
10.2. Стилль цитирования автор-год	240
 Глава 11. Предметный указатель	 246
11.1. Верстка указателей	251
11.2. Глоссарий	256
 Глава 12. Стандартные пакеты	 263
12.1. Пакеты коллекции <code>tools</code>	264
12.2. Условные конструкции пакета <code>ifthen</code>	265
12.3. Пакет <code>epstopdf</code>	267
12.4. Пакеты <code>lscap</code> и <code>pdfscape</code>	268
12.5. Пакет <code>microtype</code>	268
12.6. Гиперссылки пакета <code>hyperref</code>	269

12.7. Пакеты коллекции $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$	276
Заключение	278
Приложение А. Техническая информация	280
Приложение Б. Текстовые символы	287
Б.1. Кодировки шрифтов	287
Б.2. Текстовые символы	290
Б.3. Латиница	292
Б.4. Кириллица	296
Приложение В. Математические символы	300
В.1. Математические алфавиты и акценты	301
В.2. Греческие буквы	309
В.3. Математические символы	310
В.4. Точки и многоточия	318
В.5. Математические функции	319
Приложение Г. Генерация предметного указателя	321
Список литературы и интернет-источников	326
Предметный указатель	331
Окружения	344
Длины, счетчики, константы	346
Пакеты	350
Средства верстки библиографии	352
Список иллюстраций	355
Список таблиц	358

Алексей Владимирович Кузнецов

ОСНОВЫ L^AT_EX

Учебное пособие

Редактор М.В. Макарова
Оригинал-макет изготовлен А.В. Кузнецовым

Подписано в печать 23.04.2021. Формат 60×84 1/16.
Печ. л. 22.75. Уч.-изд. л. 22.75. Тираж 100 экз.
Изд. № 007-1. Заказ № 3.

Национальный исследовательский ядерный университет
«МИФИ».

Типография НИЯУ МИФИ.
115409, Москва, Каширское ш., 31.