Математическая типография

Авторы УМКД:

О.В. Знаменская

С.В. Знаменский

В.П. Кривоколеско

Д.Е. Лейнартас

В.М. Трутнев

1. Предыстория

Дональд Кнут создал замечательную программу TEXи нарисовал шрифты (Computern Modern). Теперь TEX'ом и программами, созданными на его основе, пользуются много людей на всей планете (список больших коллекций в последнем TEXLive включает: арабский, армянский, африканские языки (частично), британский и американский варианты английского, венгерский, вьетнамский, голландский, греческий, датский, иврит, индийские языки, испанский, итальянский, кириллические языки, китайский, японский и корейский, латынь, маньчжурские языки, монгольский, немецкий, норвежский, польский, португальский, тибетский, финский, французский, хорватский, чешский и словацкий языки, шведский).

В настоящее время практические вся математическая литература печатается в TEX'е, в TEX'е набирается значительная часть естественнонаучных книг и журналов (многие научные журналы, которые принимают рукописи

Как гласит легенда, Дональд Кнут (Donald E. Knuth) написал программу TeX для себя, чтобы готовить к печати книги серии "Искусство программирования для ЭВМ". Первая версия TeX'а появилась в 1978 году; начиная с 1989 года все изменения в TeX'е сводятся в основном к исправлению ошибок (Кнут сознательно не стремится совершенствовать TeX дальше, чтобы предотвратить распространение несовместимых версий). Тому, кто первым обнаружит очередную ошибку, Кнут выплачивает денежную премию; время от времени ставка удваивается, и тем не менее Кнут отнюдь не разорился1).

Программа TeX распространяется свободно, а ее исходные тексты полностью доступны: помимо книги [2], обязательной для изучения каждому, кто собирается стать TeX ником, в виде книги опубликован и полный текст программы с комментариями. Текст программы написан на изобре-

текст программы с комментариями. Текст программы написан на изобретенном Кнутом псевдокоде под названием Web; раньше Web транслировали в Pascal (с помощью программы, написанной все тем же Кнутом), в настоящее время используется транслятор из Web'а в язык C.

Макропакеты и форматы Если получить программу TeX из кнутовского исходного текста (то есть перевести псевдокод в Pascal или C, а затем скомпилировать), то использовать полученную программу ("чистый TeX" — по-английски говорят еще "virgin TeX") для обработки текстов будет еще невозможно: чистый TeX не знает смысла ни спецзнаков2), ни макросов. В него встроены только "примитивные команды".

Стало быть, прежде чем использовать TeX для дела, его надо обучить смыслу тех многочисленных макросов, которые используются в реальных tex-файлах. Набор TeX'овских макросов, используемый для обработки текстов, называется макропакетом. С одним из TeX'овских макропакетов вы

Как же TeX узнает определения макросов из используемого вами макропакета? Ответ таков. Среди известных чистому TeX'у примитивных команд есть и команда, предназначенная для определения макросов (чем-то похожая на известные вам `\newcommand` и `\renewcommand`); если перед началом файла с текстом и формулами, который вы хотите обработать, приписать "исходный текст макропакета" — список всех используемых в макропакете (скажем, том же LaTeX'е) макроопределений — и передать полученный файл для обработки программе TeX, то TeX сначала узнает смысл используемых вами макросов, а к моменту начала собственно текста будет уже "во всеоружии" и благополучно создаст нужный вам dvi-файл.

Первый макропакет написал одновременно с программой TeX сам Кнут: это пакет Plain TeX, который также описан в книге [2].

Следующим после Plain TeX'а макропакетом, получившим распространение, стал AMS-TeX, созданный Майклом Спиваком (Michael Spivak). По сравнению с Plain TeX'ом в него были добавлены удобные средства для набора сложных формул, особенно многострочных. Подчеркнем, что всех AMS-TeX'овских эффектов можно добиться и в Plain TeX'е — возможности TeX'а новый макропакет расширить не может, — но там это требует громоздких записей и довольно серьезного знания TeX'овских внутренних механизмов; Спивак создал для этих целей удобные сокращенные обозначения (то есть макроопределения!), которыми математики стали с удовольствием пользоваться (и до сих пор пользуются).

Структура документов в LaTeX

Структура исходного файла. Логическая структура. Команды секционирования. Структура оглавления. Базовые классы и пакеты.

Структура исходного файла проста. Она состоит из тела документа и шапки, в которой указывается, какие средства должен задействовать LaTeX, чтобы корректно обработать документ. Таким образом, шапка отвечает за стиль оформления, а тело — за содержание документа. Минимальный документ LaTeXна русском языке можно получить, скомпилировав обычный текстовый файл с расширением `.tex` и со следующими строчками:

```
\documentclass{article}
\usepackage[russian]{babel}
\begin{document}
Крошечный документ. У него нет структуры!
 \end{document}
```

После компиляции получится следующий текст:

Крошечный документ. У него нет структуры!

Класс документа — это первое, что требуется указать при наборе. Класс документа выбирается с помощью команды

```
\documentclass[необязательные параметры]{имя класса}.
```

2. Пакеты и классы документов в LaTeX

2.1. Базовые классы документов в LaTeX

Исторически LaTeXсоздавался для облегчения создания разного рода *математических* текстов. Какую печатную продукцию создает математик? В первую очередь это статьи, книги, отчеты, доклады, письма и слайды для устных выступлений. Каждый из этих документов имеет свое назначения, стиль, правила оформления. Соответственно и в LaTeX'е содержится 6 базовых классов, которые определяют вид и структуру документа, отвечающего американским полиграфическим требованиям к данному типу публикации:

2.2. Стандартные специализированные пакеты

При помощи базовых классов LaTeX можно создать документ, вполне удовлетворяющий стандартным требованиям.

Однако, как правило, при помощи базовых классов не удается или довольно сложно реализовать авторский замысел дизайна печатного (электронного) продукта, поэтому на основе базовых классов или «с нуля» создаются новые классы. Неудивительно, что количество уникальных файлов с расширением `.cls` в последних дистрибутивах LaTeXуже приближается к двум сотням.

Многие издательства и журналы предоставляют свои собственные стилевые файлы, что позволяет быстро оформить публикацию, соответствующую стандартам издания. Каждый серьёзный научный журнал и крупная конференция имеет свой LaTeX-класс. Как правило, этот класс лежит где-то на официальном сайте, например, журналы, издаваемые издательством МАИК «Наука/Интерпериодика», должны следовать правилам выложенным здесь: http://www.maik.ru/pub/tex/.

Классы документов Американского математического общества

Американское математическое общество (AMS) распространяет пакет из трех специализированных классов документа, предназначенных для набора математических текстов: `amsart`, `amsbook` и `amsproc`, которые используются для материалов, публикуемых в журналах $\mathcal{AMS}$и Сибирском математическом журнале. Все $\mathcal{AMS}$'овские классы документов автоматически подключают стилевые пакеты `amsmath` и `amsthm` для оформления теорем и печати математических символов. Основные особенности этих классов относятся к титульной информации и рубрикации документа.

Пакет классов «extsizes»

Пакет extsizes содержит классы extarticle, extbook, extletter, extproc, extreport, по назначению аналогичные классам article, book, letter, proc, report. Базовый набор пакетов не позволяет использовать базовые размеры шрифта в 9pt, 14pt и 17pt. Этот конкретный недостаток ликвидирован в пакетах классов extsizes, которые отличаются от стандартных только возможностью указать базовый размер шрифта, отличный от обычного 10-12pt.

Пакет NCC для создания русскоязычных документов

Ещё один вариант замены стандартных классов — это пакет NCC.

Для использования следует загрузить класс ncc и передать ему желаемый стиль оформления.

Класс NCC содержит четыре стиля оформления документов, заказываемые обычными опциями article (статья), preprint (препринт), book (монография) и report (отчет), который эквивалентен стилю book с опцией oneside.

Классы для создания конструкторской документации

Основу коллекции составляют три класса: `eskdtext` (для текстовой документации), `eskdbtab` (для чертежей и схем) и `eskdgraph` (для документов, разбитых на графы). Пакет снабжен понятной документацией на русском языке (`eskdx.pdf`).

2.3. Новые классы под разные задачи

Остановимся на разновидностях классов и пакетов (наборов классов), наиболее перспективных в отношении их использования в учебно-научном процессе.

Подготовка научных статей

Верстка монографий, учебников и других книг

Подготовка отчетов

Подготовка курсовых, дипломных работ и диссертаций

Подготовка презентаций

Оформление сборников и трудов конференций

Создание учебно-методических и других полезных материалов

LaTeX и поддержка многоязычной среды. Пакеты babel, ncc, eskdx, gost. Автоматические переносы слов.

В момент возникновения TEX'а его создатель Дональд Кнут, конечно не предполагал, что эта система будет популярна во всем мире. Поэтому без дополнительных усилий TEX и LATEX поддерживают издание только англоязычной типографской продукции. При попытке использовать TEX для верстки текстов на других языках, возникает ряд трудностей.

Кодовая страница (codepage) — это способ хранения информации о шрифте в памяти компьютера, включающий заранее заданный упорядоченный стандартный комплект знаков шрифта для набора текста на определенных языках.

Кодировка шрифта (encoding) представляет собой таблицу символов, где каждой букве алфавита (а также цифрам и специальным знакам) присвоен свой уникальный номер — код символа.

Кириллические кодировки появились исторически, в связи с проблемами адаптации к русскому языку разных операционных систем (DOS, MS Windows, UNIX или Mac OS). Кодовые страницы обозначались стандартным индексом или словом.

- Исторически первая и ныне основная в UNIX российская кодировка называется KOI-8.
- Для MS-DOS существует своя DOS-кодировка (кодовая страница 866).
- Стандарт для кодировки Кириллицы в Windows — кодовая страница CP1251. Де-факто, она стала самой распространенной в России.
- Для редких в России пользователей Macintosh-компьютеров была придумана еще одна кодировка — MAC.

Поддержка русского языка для LaTeX включена в MiKTeX. Для создания русскоязычных документов достаточно подключить несколько основных пакетов.

Если во входном файле встречаются символы, не входящие в стандартные 128 символов ASCII, то LaTeX'у необходимо явно указать используемую кодировку. Для восприятия LaTeX'ом кодировки служит пакет `inputenc` — он предназначен для указания кодовой страницы исходного файла. Указанный пакет подключается в преамбуле сразу после объявления класса документа:

```
\usepackage[имя_кодировки]{inputenc}
```

Можно указать сразу несколько кодировок через запятую. Заметим, что кодировкой по умолчанию станет та из них, которая указана последней.

Если необходимо поменять кодировку шрифта в теле документа, используются следующие декларации:

```
\fontencoding{кодировка шрифта}
\selectfont
```

При этом команда `\fontencoding` только объявляет кодировку, а команда `\selectfont` служит для того, чтобы LaTeX стал использовать шрифты в объявленной кодировке. Между этими командами не должно быть текста (допускаются только пробелы и декларации выбора других атрибутов шрифтов). Кодировка шрифта должна быть стандартной, иначе ее надо объявить в преамбуле посредством `fontenc`.

Пакет `russianb` также дает возможность печатать имена функций (тригонометрических, гиперболических и др.) согласно российской традиции (в скобках указаны названия функций):

```
\arctg   (arctg) \ch      (ch)  \ctg (ctg)    \sh (sh)
\arcctg (arcctg)\cosec (cosec)\cth (cth)    \tg (tg)
\th      (th)     \Prob     (P)  \nod (н.о.д.)\nok(н.о.к.)
\Proj    (Пр)     \Variance(D)  \NOD (НОД)    \NOK(НОК)
```

Команды `\nod`, `\nok`, `\NOD`, `\NOK` и `\Proj` работают только после подключения пакета `mathtext`.

В английском языке приняты одинарные и двойные кавычки в виде ‘...’ и “...”. *В России* приняты французские кавычки «елочки» и немецкие кавычки „лапки“. „Лапки“ обычно используются внутри «ёлочек», например, «...наш гордый „Варяг“...». После подключения пакета `russianb` в русском тексте возможны следующие способы набора:

- кавычки-лапки „...“ набираются "‘...’" или `\glqq` ... `\grqq`;
- кавычки-елочки «...» набираются "<...">" или `\flqq` ... `\frqq`, или <<...>>.

Активный символ " вообще не является кавычками, поэтому писать "..." — неправильно!!!

Для печати *дефиса* в составных словах введены следующие команды:

" ~ печатает дефис «-» и запрещает делать переносы в самих словах;

"= печатает дефис «-», TEX как может сам делает переносы в словах;

Чтобы получить автоматические переносы в составных словах и русское тире, соответствующее к тому же нормам и традициям российского книгопечатания, рекомендуется воспользоваться в преамбуле командой

```
\usepackage[shortcuts,cyremdash]{extdash}
```

При этом набор составных слов слегка отличается от описанного для russianb, и нужно различать в наборе следующие случаи:

\= / составное слово можно переносить по слогам, но не по дефису: σ\=/конечная и L\=/аппроксимация;

\-/ составное слово можно переносить как по слогам, так и по дефису: розово\-/фиолетовый, Римский\-/Корсаков (один человек);

\-- связанные коротким тире слова можно переносить как по слогам, так и по дефису: закон Бойля\--Мариотта (два ученых), десять\--двенадцать этажей;

\--- длинное тире, связывающее предложения или сочетания слов: и все это \--- бесплатно.

1. Классификация наборных шрифтов (Typeface classification)

Шрифт — это система визуального отображения информации при помощи условных символов. В более узком значении это комплект знаков определенного рисунка — литер, цифр и специальных знаков.

Гарнитура — это совокупность шрифтов, объединенных одним характером рисунка букв, вне зависимости от их разновидностей и размеров. Каждая гарнитура получила свое название, например, литературная, обыкновенная, академическая и т. д.

Одна гарнитура внутри группы объединяет шрифты, различные по начертанию (*курсив*, **жирное**, полужирное) и размерам (кеглю), но с одинаковым рисунком.

- Шрифты с засечками — шрифты группы Антиква (Antiqua, Serif typefaces).

Это шрифт гарнитуры Антиква!

Это шрифт гарнитуры Лазурская!

Это шрифт гарнитуры журнальная!

- Группа брусковых шрифтов.

Это шрифт гарнитуры Балтика!

- Группа рубленых (гротескных, плакатных) шрифтов (Sans Serif, Gothic).

Это шрифт гарнитуры Журнальная рубленая!

Это шрифт гарнитуры Ариал!

- Дополнительная группа.

Это шрифт гарнитуры Рекламный!

2. Выбор шрифтов и их атрибутов в LaTeX

В LaTeX'е доступны разные виды текстовых шрифтов, которые характеризуется следующими четырьмя атрибутами: семейством (`family`); насыщенностью (`series`); начертанием (`shape`); размером (`size`).

Семейство шрифта выражает примерно то же самое, что и группы гарнитур (с засечками, без засечек и т. д.). В стандартной комплектации LaTeXопределены три семейства: шрифты с засечками, шрифты без засечек и шрифты типа «пишущая машинка».

Насыщенность определяет ширину и жирность шрифта. В стандартной комплектации LaTeXвозможны два варианта насыщенности: средняя и **полужирная**.

Кроме того, предусматривается четыре варианта начертания: прямое начертание, *курсивное начертание*, *наклонное начертание* и НАЧЕРТАНИЕ «КАПИТЕЛЬ».

Семейство, насыщенность, начертание и размер шрифта могут различным образом сочетаться. Каждый из шрифтовых атрибутов можно менять независимо от остальных.

Команды для изменения атрибутов шрифта существуют в LaTeX'е в двух вариантах: *в виде команды без аргумента и команды с аргументом*.

Для первых трех атрибутов шрифта (кроме размера) имя команды без аргумента совпадает с английским названием соответствующего атрибута:

`\rmfamily` — шрифты с засечками;
`\sffamily` — шрифты без засечек;
`\ttfamily` — шрифты типа «печатная машинка»;

`\mdseries` — средняя насыщенность;
`\bfseries` — полужирный;

`\upshape` — прямое начертание;
`\itshape` — курсив;
`\slshape` — наклонное начертание;
`\scshape` — капитель.

Имя команды с аргументом состоит из слова `text`, к которому добавлены две буквы, описывающие атрибут, например, `\textbf{...}` или `\textsc{...}`.

Лекция 4: Управление шрифтовыми выделениями и подключение шрифтов

```
Теперь мы можем сделать вид, что
\texttt{печатаем на печатной машинке},
затем шрифт с засечками заменить на
\textsf{шрифт без засечек}, после чего
\sffamily подчеркнуть важность этого текста
\textit{курсивным} или \textbf{полужирным шрифтом}
\textnormal{и, наконец попробовать}
\textbf{\textit{соединить три в одном}}.
```

Теперь мы можем сделать вид, что `печатаем на печатной машинке`, затем шрифт с засечками заменить на шрифт без засечек, после чего подчеркнуть важность этого текста *курсивным* или **полужирным шрифтом** и, наконец попробовать ***соединить три в одном***.

Специальная команда `\emph{...}` позволяет набирать выделяемую фразу курсивом, если текущий шрифт прямой и и прямым шрифтом, если текущий шрифт наклонный. В обычном тексте, набранном прямым шрифтом, результаты действия команд `\emph{...}` и `\textit{...}` неразличимы.

Обратим внимание на то, что команда `\textnormal` сразу возвращает к изначальному шрифту, который был по умолчанию. Вариант этой команды без аргумента `\normalfont` работает также, т.е. переключает шрифт на основной шрифт документа.

— Терем-теремок!

Кто в тереме живет?

— Я, мышка-норушка.

— Я, лягушка-квакушка.

— Я, зайчик-побегайчик.

— Я, лисичка-сестричка.

— Я, волчек-серый бочек.

— А ты кто?

— А я медведь косолапый.

— А я медведь косолапый.

Изменение размера шрифта определяется командами без аргумента. Всего их десять и все они использованы в следующем примере.

```
{\tiny          -- Терем-теремок! }        \\
{\scriptsize      Кто в тереме живет?}    \\
{\footnotesize -- Я, мышка-норушка.}      \\
{\small         -- Я, лягушка-квакушка.}  \\
{\normalsize    -- Я, зайчик-побегайчик.} \\
{\large         -- Я, лисичка-сестричка. } \\
{\Large         -- Я, волчек-серый бочек.} \\
{\LARGE         -- А ты кто?}              \\
{\huge          -- А я медведь косолапый.} \\
{\Huge          -- А я медведь косолапый.}
```

Добавление одного шрифта

```
\newfont{\команда}{описание_шрифта}
```

```
\newfont{\strannii}{cmff10}
```

Strange font: strange, strange, strange

```
Strange font: {\strannii strange, strange, strange}
```

```
\newfont{\wasy}{wasy10}
\newfont{\bigwasy}{wasy10 scaled 2000}
\newfont{\largewasy}{wasy10 at 40pt}
\newcommand{\bs}{\wasy\symbol{45}}
\newcommand{\bbs}{\bigwasy\symbol{45}}
\newcommand{\lbs}{\largewasy\symbol{45}}
```

wasysym

\blacksmiley

```
\bs \bbs \lbs \lbs \bbs \bs
```

Команды для подключения семейства шрифтов

```
\usepackage{times}
\usepackage{mathptmx}
```

```
\fontfamily{xxx}
\fontseries{...}
\fontshape{...}
\fontsize{...}{...}
\selectfont
```

3. Коллекции кириллических шрифтов (Cyrillic fonts) в LaTeX

Кириллические шрифты LH

`lcmss` Рубленый шрифт для слайдов (slides sans serif), прямой, наклонный (slanted), полужирный (bold);

`lcmtt` Шрифт пишущей машинки для слайдов (slides typewriter), прямой, курсив (italic), невидимый шрифт пишущей машинки;

`lcmss` Рубленый шрифт для слайдов (slides sans serif), прямой, наклонный (slanted), полужирный (bold), невидимый рубленый шрифт для слайдов (invisible ss), невидимый наклонный (invisible slanted);

`cmd` Данхилл (Danhill), одно начертание;

`cmfib` Фибоначчи (Fibonacci), прямой и наклонный (slanted);

`cmfr` Забавный (funny roman), прямой и курсив (italic);

`cmvtt` Разноширинный шрифт пишущей машинки (variable typewriter), прямой и курсив (italic);

`cmbr` Шрифт семейства CM Bright, прямой, наклонный (slanted), полужирный (semibold, bold);

`cmtl` Шрифт семейства CM Bright Typewriter, прямой и наклонный (slanted);

`ccr` Шрифт для «Конкретной математики».

Лекция 4: Управление шрифтовыми выделениями и подключение шрифтов

```
\fontfamily{ccr}
\mdseries\upshape
Шрифт для «Конкретной математики»(Concrete roman):

\slshape наклонный (slanted), \itshape курсив (italic),
\scshape капитель (small caps), \bfseries\upshape
полужирный (bold), \itshape полужирный курсив (bold
italic), \slshape полужирный наклонный (bold slanted),
\scshape полужирная капитель (bold small caps),
\fontseries c\slshape суженный наклонный
(condensed slanted).
```

Шрифт для «Конкретной математики»(Concrete roman):

наклонный (slanted), *курсив (italic)*, КАПИТЕЛЬ (SMALL CAPS), **полужирный (bold)**, ***полужирный курсив (bold italic)***, ***полужирный наклонный (bold slanted)***, **полужирная капитель (bold small caps)**, *суженный наклонный (condensed slanted)*.

Забавный шрифт хорош для таких забавных текстов, как этот:

Варкалось, хливкие шорьки
Пырялись по наве,
И хрюкотали зелюки,
Как мюмзики в мове.

Шрифт Фибоначчи, *увы, имеет лишь* прямое *и наклонное начертания*.

```
\fontfamily{cmfib}\mdseries\upshape
Шрифт Фибоначчи, \textsl{увы, имеет лишь
\emph{прямое}  и наклонное начертания}.
```

Пакет CM-Super — коллекция PostScript-шрифтов

PSCyr — коллекция PostScript-шрифтов

```
\textcp{Кладезь (CooperPSCyr), одно начертание}
\textad{Рекламный (AdvertisementPSCyr), одно начертание}
```

- Академическая (AcademyPSCyr); начертания: Прямое нормальное, **Полужирное**, *Курсивное*
- Антиква (AntiquaPSCyr); начертания: Прямое нормально **Полужирное**, *Курсивное*, ***Полужирное Курсивное***
- Балтика (HandbookPSCyr); начертания: Прямое нормальное, **Полужирное**, *Курсивное*
- Бодони (CollegePSCyr); начертания: Прямое нормальное, **Полужирное**, *Курсивное*

- **Кладезь (CooperPSCyr), одно начертание**
- **Рекламный (AdvertisementPSCyr), одно начертание**

- Журнальная (JournalPSCyr); начертания: Прямое нормальное, **Полужирное**, *Курсивное*
- Лазурская (Lazurski); начертание: Прямое нормальное
- Сувенир (SouvenirPSCyr); начертания: Прямое нормальное, **Полужирное**
- Таймс (TimesNewRomanPSMT); начертания: Прямое нормальное, **Полужирное**, *Курсивное*, ***Полужирное Курсивное***

- Курьер (CourierNewPSMT); четыре начертания: Прямое нормальное, **Полужирное**, *Курсивное*, ***Полужирное Курсивное***
- ER Курьер (ERKurierPSCyr); четыре начертания: Прямое нормальное, **Полужирное**, *Курсивное*, ***Полужирное Курсивное***

URWCyr — коллекция кирилизированных шрифтов

А. Шрифты с засечками. В эту группу входят гарнитуры URW Bookman; Century Schoolbook; URW Palladio; Nimbus Roman №9. Их аббревиатуры, соответственно, `ubk`, `unc`, `upl`, `utm`.

Б. Шрифты без засечек. Названия шрифтов: URW Gothic и Nimbus Sans, аббревиатуры, соответственно, `uag` и `uhv`.

В. Имитация шрифтов пишущей машинки. Шрифт называется Nimbus Mono и имеет внутреннее имя (аббревиатуру) `ucr`.

```
\renewcommand{\rmdefault}{aaa}
\renewcommand{\sfdefault}{bbb}
\renewcommand{\ttdefault}{ucr}
```

Загадка

Долго я следил за ней.
А она с утра до ночи
то сама себя длинней,
ТО САМА СЕБЯ КОРОЧЕ.

Формулы в TEX'е делятся на два вида: формулы *внутри текста* и *выключные*, т. е. вынесенные в отдельную строку.

Внутритекстовые формулы окружаются с обеих сторон знаками $ или заключаются между `\begin{math}` и `\end{math}`. TEX автоматически сжимает внутритекстовые формулы так, чтобы они заняли как можно меньше места по высоте. Для математических символов, функций, операций и операторов определены специальные команды LATEX'а. Например, текст:

```
Истинный математик~-- $\tau\epsilon\chi$-нолог
отдает свое $\heartsuit$ математической моде,
в которой легко написать, что
\begin{math}\sin^2 \alpha+\cos^2 \alpha=1\end{math}.
```

после компиляции будет иметь вид:

Истинный математик – $\tau\epsilon\chi$-нолог отдает свое $\heartsuit$ математической моде, в которой легко написать, что $\sin^2 \alpha+\cos^2 \alpha=1$.

Выключные формулы окружаются знаками $$ или парой команд \[и \] (или заключаются между \begin{displaymath} и \end{displaymath}). Использование команд \[и \] позволяет одной стилевой опцией во всём документе прижать выключные формулы влево, тогда как формулы, заключённые между $$ и $$, всегда центрированы.

Формулы нельзя разрывать пустой строкой.

Выключные формулы TEX печатает с временным прерыванием текущего абзаца, поэтому когда после формулы нет пустой строки или команды \par, следующая за формулой строка печатается без абзацного отступа.

В стандартных классах документа по умолчанию формулы располагаются по центру строки. Включение опции fleqn в команде \documentclass[fleqn]{article} выравнивает все формулы по левому краю страницы. При этом расстояние от левого поля страницы до формулы задаётся параметром \mathindent (по умолчанию 2.5 em), меняя значение которого командами \setlength или \addtolength, можно устанавливать любую величину отступа формулы от края страницы.

Дополнительный набор разнообразных символов можно найти в пакетах `txfonts` (гарнитура Times) и `pxfonts` (гарнитура Adobe Palatino). Они содержат как экзотические (типа ♡), так и достаточно распространенные символы. Указанные пакеты переопределяют все команды для набора символов из пакетов `latexsym`, `amssymb` и `amsfonts` (математические алфавиты). Поэтому после загрузки пакета `txfonts` можно использовать символы из `latexsym`, `amssymb` и `amsfonts` не загружая сами пакеты. В частности, пакет `txfonts` содержит команды `\varg`, `\varv`, `\varw` и `\vary` для печати латинских букв g, v, w и y с другой формой символов. Чтобы изменить начертание этих букв во всем тексте, достаточно указать опцию `varg` при подключении пакета: `\usepackage[varg]{txfonts}`.

Различные математические, физические и технические символы содержатся также в пакетах `stmaryrd` и `wasysym`. В `wasysym` можно найти астрономические и астрологические знаки.

Для набора кратных интегралов имеются специальные команды. Набор при помощи `\int\int\int\int}` дает некрасивые результаты $\int\int\int\int$. Действительно:

```
\[
  \int\int\int f(x)\,dx
  \text{  --- это некрасиво!!!}
\] \[\iint\limits_X f(x)\,dx   \quad
     \iiint\limits_X f(x)\,dx  \quad
     \iiiint\limits_X f(x)\,dx \quad
     \idotsint\limits_X f(x)\,dx
   \]
```

$$\int\int\int f(x)\,dx \text{ — это некрасиво!!!}$$

$$\iint\limits_X f(x)\,dx \qquad \iiint\limits_X f(x)\,dx \qquad \iiiint\limits_X f(x)\,dx \qquad \int\cdots\int\limits_X f(x)\,dx$$

Команды `\overset` и `\underset` позволяют поместить символы меньшего размера над и под формулой, а команда `\sideset` — поместить символы меньшего размера со всех сторон большого оператора:

```
\[
\overset{*}{X} \qquad \underset{*}{X} \qquad
\underset{i=1}{\overset{\infty}{X_i}}\qquad
\sideset{_1^2}{_3^4} \prod_k \qquad
\sideset{}{'}\sum_{0\le i\le m}E_i\beta x
\]
```

$$\overset{*}{X} \qquad \underset{*}{X} \qquad \underset{i=1}{\overset{\infty}{X_i}}\qquad {}_1^2\prod_k{}_3^4 \qquad \sum_{0\le i\le m}E_i\beta x$$

Команды `\xrightarrow` и `\xleftarrow` позволяют создавать растяжимые стрелки с надписями:

```
$X \xrightarrow [B\in\mathfrak{M}^0,C\in\mathfrak{M}^1]
{A = C\circ B} Y$\qquad $Y \xleftarrow{A = C\circ B} X$
```

$$X \xrightarrow[B\in\mathfrak{M}^0,C\in\mathfrak{M}^1]{A=C\circ B} Y \qquad Y \xleftarrow{A=C\circ B} X$$

Команды `\underbrace` и `\overbrace` позволяют печатать фигурные скобки, соответственно, под и над формулой. Немного сложнее изобразить фигурные скобки и под, и над формулой:

```
$\underbrace{\overbrace{0 1 2 \dots 9}^{10}
\rm A B \dots F}_{16}$
```

$$\underbrace{\overbrace{012\ldots9}^{10}\mathrm{AB}\ldots\mathrm{F}}_{16}$$

В математической моде используются символы трех размеров: текстового, индексов и индексов к индексам. Если размер шрифта документа равен 10 pt, то размер индексов и индексов к индексам будет, соответственно, 7 pt и 5 pt. Размер символов, которыми печатается та или иная часть формулы, задаётся её стилем.

TEX сам выбирает, какой стиль использовать в числителе и знаменателе дробей, в индексах и т.д., как например, при печати этой формулы:

```
\[
\sum_{i = 1}^{ \left[ \frac{n}{2} \right] }
\binom{ x_{i, i + 1}^{i^{2}} }
{ \left[ \frac{i + 3}{3} \right] }
\frac{ \sqrt{ \mu(i)^{ \frac{3}{2}} (i^{2} - 1) } }
{ \sqrt[3]{\rho(i) - 2} + \sqrt[3]{\rho(i) - 1} }
\]
```

$$\sum_{i=1}^{\left[\frac{n}{2}\right]} \binom{x_{i,i+1}^{i^2}}{\left[\frac{i+3}{3}\right]} \frac{\sqrt{\mu(i)^{\frac{3}{2}}(i^2-1)}}{\sqrt[3]{\rho(i)-2}+\sqrt[3]{\rho(i)-1}}$$

Стиль формулы можно изменить вручную следующими декларациями:

Перекрытие фигурных скобок требует преодоления TEX-нических трудностей, поскольку группы не могут перекрываться. Команда `\phantom` производит невидимую формулу, над которой рисуется скобка. Команда `\lefteqn` декларирует, что эта формула не занимает места по ширине:

```
$\lefteqn{\underbrace{\phantom{a_1+\dots+a_i+a_j}}}
a_1+\dots+\overbrace{a_i+a_j+\dots+a_n}$
```

$$\underbrace{a_1+\cdots+}\overbrace{a_i+a_j+\cdots+a_n}$$

Команда `\substack` создает «многоэтажные» индексы:

```
\[
\sum_{\substack{n_1, n_2, \dots, n_r \\
n_1+n_2+\dots+n_r=n\\n_1,n_2,\dots,n_r>0}}
\frac{n!}{n_1!\,n_2!\dots n_r!}
\]
```

$$\sum_{\substack{n_1,n_2,\dots,n_r\\ n_1+n_2+\dots+n_r=n\\ n_1,n_2,\dots,n_r>0}} \frac{n!}{n_1!\,n_2!\dots n_r!}$$

1. Оформление ссылок на литературу в исходном файле

```
\begin{thebibliography}...\end{thebibliography}
```

```
\addcontentsline{toc}{chapter}{Литература}
\begin{thebibliography}{99}
\bibitem{metkacite} выходные данные публикации 1.
    <...>
\bibitem{metkacite} выходные данные публикации  n.
```

Конструкция означает, что глава «Литература» добавляется в конец списка оглавления, а ссылки (bibitem) указываются в порядке перечисления их ниже.

Рассмотрим в качестве примера документ класса `article`, в котором ссылки на литературу нумеруются в порядке упоминания.

```
\documentclass[a4paper,12pt,oneside]{article}
\usepackage[T2A]{fontenc}
\usepackage[cp1251]{inputenc}
\usepackage[english,russian]{babel}
\usepackage{amssymb,amsfonts,amsmath,
mathtext,cite,enumerate,float}
\begin{document}
<...>
Метод Голда\cite{gold} получен из метода
ван Циттерта\cite{gold-cittert},
но с предположением, что действует он
только для положительных чисел.
<...>
```

Лекция 6: Библиография и цитирование литературы

```
\begin{thebibliography}{8}% Список литературы
\bibitem{blind}
S. M. Jefferies and J. C. Christou.
Restoration of astronomical images by iterative
blind deconvolution.
Astrophysical Journal, 415:862+, October 1993.
<...>
\bibitem{gold}
GOLD, R.: An Iterative Unfolding Method for
Matrices, Tech. Rep. ANL-6984,
Argonne National Laboratory, Argonne, Illinois, 1964.
\bibitem{gold-cittert}
Journal of Electrical Engineering,
Vol. 56, NO. 5-6, 2005, 141--145
Multidimensional FFT Based Positive
Definite Gold Deconvolution Algorithm
Miroslav Morh\'a\v{c} -- Vladislav Matou\v{s}ek
<...>
\end{thebibliography}
\end{document}
```

В этом случае ссылки `\cite{metkacite}` будут пронумерованы в том порядке, в котором они упомянуты в окружении `\begin{thebibliography}` ... `\end{thebibliography}`, но не в порядке упоминания в тексте.

Метод Голда [4] получен из метода ван Циттерта [5], но с предположением, чт
чисел.

«. . . »

Список литературы

[1] S. M. Jefferies and J. C. Christou. Restoration of astronomical images by
Journal, 415:862+, October 1993.

[2] Van CITTERT, P. H. : Zum Einfluss der Spaltbreite auf die Intensittverteilu
69 (1933), 298.

[3] N.R. Hill, G.E. Ioup: J. Opt. Soc. Am. 66 (1976) 487

[4] GOLD, R. : An Iterative Unfolding Method for Matrices, Tech. Rep. ANL-6
Illinois, 1964.

[5] Journal of Electrical Engineering, Vol. 56, NO. 5-6, 2005, 141–145 Multidin
Deconvolution Algorithm Miroslav Morháč – Vladislav Matoušek

[6] L.B. Lucy. An iteration technique for the rectification of observed distributic

2. Создание списка литературы с использованием библиографической базы данных BiBTeX

Заметим, что если при работе с большими списками цитируемой литературы случайно переставить местами в окружении `\begin{thebibliography}` ... `\end{thebibliography}` ссылки `\bibitem{gold}` и `\bibitem{gold-cittert}`, то в тексте ссылки будут стоять так:

Такая особенность затрудняет работу с большими документами, а также с документами, составленными из нескольких файлов. Особенно ярко этот недостаток проявляется тогда, когда часть документа с цитированием литературных источников вставляется в другой документ.

Сократить время формирования списка и избежать указанных трудностей позволяет создание собственной, периодически обновляемой библиографической базы данных автора. Источники литературы в этом случае добавляются не в сам LaTeX-документ, а во вспомогательный `bib`-файл, что позволяет использовать созданную библиографическую базу для многих документов.

Библиографические базы имеют определенную структуру. База хранится в обычном текстовом файле с расширением `.bib` (в нем перечисляются все библиографические ссылки автора). В процессе работы с файлом `имяфайла.tex` программа `BibTex` извлекает из базы данных источники литературы, указанные в тексте файла.

Преимущества использования библиографических баз данных

1. Ссылки в основном документе появляются по мере упоминания и всегда нумеруются правильно.
2. Существует возможность использования одной базы во многих документах.
3. Если в описание ссылки на литературу вкралась ошибка, её можно исправить в базе, тем самыи она будет исправлена сразу во всех ссылающихся на базу документах.
4. Существует простой поиск в базе для выяснения ключа цитирования (аргумента `\cite{metkacite}`).
5. Возможен импорт и экспорт в другие библиографические форматы.

Порядок формирования и работы с личной базой данных

1) Наличие необходимых файлов с расширениями `.bib` и `.bst`.

2) Команды в основном файле статьи (или другого научного труда).

В преамбуле статьи нужно разместить команду `\bibliographystyle{библиографический стиль}`, аргумент которой — это имя используемого `.bst`-файла `библиографический стиль.bst`.

В конце статьи вместо окружения `\thebibliography` нужно разместить команду `\bibliography{имя файла базы данных}`, где имя файла указывается без расширения.

Порядок формирования и работы с личной базой данных

3) Последовательность действий при работе с подготовленным файлом статьи и базой данных.

- Статья обрабатывается один раз при помощи latex (pdflatex). В результате этого в файле с расширением `.aux` генерируются `\cite` ссылки статьи.

- В командной строке (например в `Far`) дается команда `bibtex имястатьи`, где имя файла статьи указывается без расширения. При этом, на основании анализа файла `.aux`, `BibTeX` создает файл `.bbl`, содержащий ссылки, отформатированные в соответствии с библиографическим форматом, определенном в стилевом файле `.bst`. Предупреждения и сообщения об ошибках записываются в файл `.blg`.

Порядок формирования и работы с личной базой данных

3) Последовательность действий при работе с подготовленным файлом статьи и базой данных.

- Теперь LaTeX должен считать файл `.bbl` (статья еще раз обрабатывается при помощи latex).
- Чтобы ссылки в тексте приняли окончательный вид, статья третий раз обрабатывается при помощи latex (pdflatex).

Полезные советы

```
\makeatletter
\bibliographystyle{unsrt}
\renewcommand{\@biblabel}[1]{#1.}
\makeatother
```

```
\bibliography{biblio/my,biblio/your}
```

`biblio` – имя папки в рабочей директории

Лекция 6: Библиография и цитирование литературы

Создание списка цитируемой литературы и оформление на него ссылок является, пожалуй, самым трудоемким из технических процедур подготовки научной статьи и объемных научных трудов.

В LaTeX'е более-менее автоматизированно это можно сделать двумя способами.

Ссылки на литературу можно создать при помощи окружения `\begin{thebibliography}...\end{thebibliography}`. В серьезных научных трудах библиография оформляется в виде отдельной главы:

```
\addcontentsline{toc}{chapter}{Литература}
\begin{thebibliography}{99}
\bibitem{metkacite} выходные данные публикации 1.
    <...>
\bibitem{metkacite} выходные данные публикации  n.
```

Лекция 6: Библиография и цитирование литературы

Библиографические базы имеют определенную структуру. База хранится в обычном текстовом файле с расширением `.bib` (в нем перечисляются все библиографические ссылки автора). В процессе работы с файлом `имяфайла.tex` программа `BibTex` извлекает из базы данных источники литературы, указанные в тексте файла.

Преимущества использования библиографических баз данных

1. Ссылки в основном документе появляются по мере упоминания и всегда нумеруются правильно.
2. Существует возможность использования одной базы во многих документах.
3. Если в описание ссылки на литературу вкралась ошибка, её можно исправить в базе, тем самыи она будет исправлена сразу во всех ссылающихся на базу документах.
4. Существует простой поиск в базе для выяснения ключа цитирования (аргумента `\cite{metkacite}`).
5. Возможен импорт и экспорт в другие библиографические форматы.

Порядок формирования и работы с личной базой данных

1) Наличие необходимых файлов с расширениями `.bib` и `.bst`.

Как уже отмечалось, сама база данных находится в файле с расширением `.bib`. Перед началом необходимо создать и заполнить файл с таким расширением.

Каждая запись должна быть отнесена к определенному типу: статья в журнале, статья для конференции, книга, дипломная работа и так далее.

Все записи файла `имяфайла.bbl` должны иметь определенную структуру. Записи различных типов в базе данных имеют различный набор полей, причем поля делятся на 3 класса: обязательное поле, необязательное поле и игнорируемое поле. Подробно ознакомиться со списком полей для различных типов записей можно в документации по BiBTeXу.

Синтаксис описания элементов указателя. Подготовка указателя. Программа MakeIndex.

1. Управление заголовками

1.1. Построение алфавитного указателя

В этой разделе мы рассмотрим составление алфавитного указателя. Уже из его названия ясно, что в таком указателе список терминов упорядочен по алфавиту. LaTeX предоставляет средства, позволяющие полностью автоматизировать составление алфавитного указателя и значительно облегчить подготовку словаря терминов. Текст алфавитного (равно как и любого другого) указателя печатает процедура `theindex`.

Подготовка алфавитного указателя — весьма длительная процедура. В качестве первого шага в исходном тексте следует пометить выбранные термины при помощи команды `\index`. Если некоторые термины встре-

Чтобы пометить термин в исходном тексте документа в простейшем случае его достаточно продублировать в аргументе команды `\index`, которую размещают сразу вслед за термином.

На завершающей стадии подготовки указателя в корневой входной файл (допустим, он называется `my.tex`) нужно внести следующие изменения: 2. 3. 4.

1. Загрузить пакет `makeidx` и вставить в преамбулу декларацию `\makeindex`. Она указывает, что LaTeX должен обновлять перечень помеченных терминов при очередной компиляции исходного текста.

```
\documentclass{book}
\usepackage[russian]{babel}
\usepackage{makeidx}
\makeindex
. . .
\begin{document}
. . .
```

2. Вставить команду `\printindex` там, где должен быть напечатан алфавитный указатель. Обычно её помещают ближе к концу документа.

3. Повторить компиляцию входного файла. В результате будет создан первичный указатель `my.idx`. Он состоит из входов в указатель, каждый из которых создан одной командой `\index` в исходном тексте.

4. Отсортировать первичный указатель программой `MakeIndex`. В результате будет создан файл my.ind, содержащий исходный текст алфавитного указателя в виде процедуры `theindex`.

При следующей компиляции исходного текста команда `\printindex` напечатает алфавитный указатель, взяв его текст из `my.ind`.

LaTeX определяет, какие разделы документа нумеровать, а какие — нет, исходя из двух вещей: *уровня вложенности раздела* и значения счетчика `secnumdepth`. Раздел документа получает номер, если уровень его вложенности меньше или равен значению `secnumdepth` (разумеется, все сказанное относится к случаю, когда LaTeX'овская команда для раздела документа дана без звездочки — иначе нумерации не будет заведомо). Уровни вложенности в стандартных стилях приведены в таблице.

Название раздела	Уровень
`\section`	1
`\subsection`	2
`\subsubsection`	3
`\paragraph`	4
`\subparagraph`	4

Стало быть, если мы хотим, чтобы самый мелкий из нумеруемых разделов был `\paragraph`, то надо написать в преамбуле документа команду

```
\ setcounter{secnumdepth}{3}
```

2.2. Модификация оглавлений

Поясним как это происходит. Оглавление, списки рисунков и таблиц печатают соответствен- но команды

```
\tableofcontents
\listoffigures
\listoftables
```

Они считывают необходимую информацию из файлов, имена которых совпадают с именем входного файла (точнее, корневого входного файла), а расширения определяются из следующей таблицы:

команда:	\tableofcontents	\listoffigures	\listoftables
расширение:	toc	lof	lot

Команды `\addcontentsline` и `\addtocontents` позволяют редактировать служебные файлы с расширением `toc`, `lof` и `lot`, если необходимо осуществить тонкую настройку, например при ручном регулировании линии переноса на новую страницу части длинной таблицы.

Вы хотите в каком-то месте оглавления провести горизонтальную линейку шириной во всю страницу, то можно написать

```
\addtocontents{toc}{\hrule}
```

и в оглавлении появится линейка.

Пусть Вы не хотите, чтобы какая-то из строк в оглавлении начинала новую страницу. Тогда надо перед командой, порождающей эту строку оглавления (обычно таковой будет команда наподобие `\section`) написать в своем файле:

```
\addtocontents{toc}{\protect\nopagebreak}
```

в `toc`-файл запишется команда , и нежелательный разрыв страницы в оглавлении будет предотвращен. Если забыть про `\protect`, то получили бы весьма непонятное сообщение об ошибке.

Если необходимо озаглавить по другому само оглавление, списки рисунков или таблиц, то необходимо переопределить команды, хранящие названия оглавления, списка рисунков и таблиц:

```
\contentsname
\listfigurename
\listtablename
```

Например, мы можем переопределить список фигур:

```
\renewcommand{\listfigurename}{Список графиков}
```

Все символы, с которыми сталкивается TEX в процессе обработки входного файла, подразделены на 16 категорий с номерами от 0 до 15.

Номер	*Значение*	*Символ по умолчанию*
0	Начало команды	\
1	Начало группы	{
2	Конец группы	}
3	Математический ключ	$
4	Табулятор	&
5	Конец строки	<return>
6	Параметр	#

Номер	*Значение*	*Символ по умолчанию*
7	Верхний индекс	^
8	Нижний индекс	_
9	Игнорируемый символ	<null>
10	Пробел	
11	Буква	A-Z, a-z, А-Я, а-я
12	Другой символ	все, кроме перечисленных
13	Активный символ	~
14	Комментарий	%
15	Ошибочный символ	<delete>

Категорию любого символа можно изменить, однако следует помнить, что форматы TEX используют некоторые символы для специальных целей.

Изменить категорию символа можно командой

```
\catcode'\<символ>=<число>
```

В результате символу <символ> будет присвоен номер категории <число>. Как мы увидим далее, сочетание '\ имеет специальное значение, и заменяется кодом следующего за ней символа.

Не смотря на то, что команды рассматриваются как цельные элементы, существует возможность преобразовать их в список символьных элементов, используя

```
\strings\<команда>
```

где <команда> — это имя какой-либо команды.

Категорию любого символа можно изменить, однако следует помнить, что форматы TEX используют некоторые символы для специальных целей. Если нужно использовать какой-либо из этих символов в тексте так, чтобы он появился в выходном файле, то для этого определены специальные команды, такие как

```
\$, \%, \&, \#, \_
```

в обычном режиме, и

```
\{, \}, \backslash
```

в математическом. Команды \^ и \~ производят акценты над буквами.

Когда TEX читает строку текста из файла, он преобразует этот текст в список элементов. Элемент — это либо символ с номером категории, либо команда. Команда рассматривается как один объект, который больше не является последовательностью символов. Пробелы после команд в списке элементов не игнорируются так, как это делается во входном фай-

Кодировка, которую использует TEX для внутреннего представления символов, также основана на ASCII, и шрифты TEX устроены так, что коды тех символов, которые есть в ACSII, остаются такими же и в шрифте. Однако шрифт содержит 256 символов, тогда как отображаемых символов ASCII заведомо меньше. Кроме того, многие из символов отсутствуют на клавиатуре. Для того, чтобы в выходном документе появился такой символ, используется команда `\char`, которая позволяет обращаться к символам шрифта по их кодам. Формат вызова этой команды такой:

```
\char<число>
```

где `<число>` — это любое число от 0 до 255 с необязательным пробелом после него. `<число>` может быть записано как в привычной десятичной системе счисления, так и в восьмеричной или шестнадцатеричной. Чтобы указать TEX что число нужно рассматривать как восьмеричное, перед ним ставится одинарная кавычка, например `'37`, перед шестнадцатеричным числом ставится двойная кавычка, `"5A`.

Перейти в обратную сторону, от списка символьных элементов к элементу-команде можно посредством

```
\csname<элементы>\endcsname
```

<элементы> могут в том числе включать и другие команды, однако только такие, которые будут заменены на символы, а не на примитивы TeX. О том, каким образом обрабатываются команды, пойдет речь ниже.

Еще две команды, производящие элементы, это команды

```
\number<число>
\romannumeral<число>
```

которые заменяют <число> на его эквивалент в десятичной или римской системе счисления соответственно.

```
\uppercase<список элементов>
\lowercase<список элементов>
```

преобразуют <список элементов> в верний или нижний регистр соответственно. Осуществляется это засчет того, что у каждого символа кроме его кода и категории существует еще пара кодов — \uccode и \lccode. При переводе символа в верхний регистр его код заменяется на \uccode, при переводе в нижний — на \lccode, если только результирующий код не равен нулю. В противном случае код символа остается без изменений.

Кодировка, которую использует TEX для внутреннего представления символов, также основана на ASCII, и шрифты TEX устроены так, что коды тех символов, которые есть в ACSII, остаются такими же и в шрифте. Однако шрифт содержит 256 символов, тогда как отображаемых символов ASCII заведомо меньше. Кроме того, многие из символов отсутствуют на клавиатуре. Для того, чтобы в выходном документе появился такой символ, используется команда `\char`, которая позволяет обращаться к символам шрифта по их кодам. Формат вызова этой команды такой:

```
\char<число>
```

где `<число>` — это любое число от 0 до 255 с необязательным пробелом после него. `<число>` может быть записано как в привычной десятичной системе счисления, так и в восьмеричной или шестнадцатеричной. Чтобы указать TEX что число нужно рассматривать как восьмеричное, перед ним ставится одинарная кавычка, например `'37`, перед шестнадцатеричным числом ставится двойная кавычка, `"5A`.

Для ручного управления относительным размещением объектов на странице иногда необходимо указывать расстояния между ними. Измеряя расстояния, TEX использует понятие размера. Это понятие пришло из традиционной типографии, и для задания размеров используются специальные единицы измерения, принятые в издательстве. Однако размеры можно задать и в метрической системе. Используемые в TEX единицы измерения следующие:

pt	point	пункт
pc	pica	пика (1 pc = 12 pt)
in	inch	дюйм (1 in = 72.27 pt)
bp	big point	большой пункт (72 bp = 1 in)
cm	centimeter	сантиметр (2.54 cm = 1 in)
mm	millimeter	миллиметр (10 mm = 1 cm)
dd	didot point	дидот-пункт (1157 dd = 1238 pt)
cc	cicero	цицеро (1 cc = 12 dd)
sp	scaled point	суперпункт (65536 sp = 1 pt)

Необходимо всегда учитывать, что размер для TEX это не просто число, указывая размер обязательно нужно указывать и единицы измерения, даже если размер равен нулю. В общем случае размер задается так: [<знак>]<число><единица измерения>.

Необязательный параметр <знак> используется для задания отрицательных размеров, <число> может быть целым или дробным и задает числовое значение размера, <единица измерения> одна из перечисленных выше. Максимальное абсолютное значение размера составляет 2^30 sp, т.е. около 6 метров.

TEX измеряет все расстояния именно размерами, большую часть из которых задавать явно не приходится. Существует способ применить коэффициент растяжения (или сжатия) ко всем размерам в документе. Делается это с помощью команды

```
\magnification=<число>,
```

где <число> задает величину коэффициента, на который будут умножены все размеры, который ровно в 1000 раз меньше чем <число>. Т.е.

```
\magnification=1200
```

указывает растянуть все размеры в 1.2 раза. Конечно, чтобы TEX мог сформировать масштабированный документ в системе должны присутствовать шрифты подходящего размера.

Определять собственные регистры размера можно командой

```
\ newdimen\<name>
```

После этого можно использовать \<name> как регистр размера. Создавая собственные регистры необходимо помнить, что фактически TEX не создает нового регистра, он выбирает один из 256, которые не используется в данный момент, и назначает ему символическое имя, как было показано выше.

С регистрами размера можно выполнять арифметические операции, используя команды

```
\advance \<name> by <размер>
\multiply\<name> by <размер>
\divide  \<name> by <размер>
```

Регистр <name> будет использован в качестве левого операнда, а <размер> в качестве правого для операции сложения, умножения или деления соответственно. Результат операции будет помещен в тот же регистр.

С точки зрения подхода к TEX как к языку программирования, регистры размера больше всего походят на типизованные переменные. В сущности, таковыми они и являются, наряду с регистрами счетчиков, клея и боксов, о которых еще пойдет речь.

Лекция 9: Счетчики, длины, боксы

Когда TEX имеет дело с потоком элементов (напомним, что поток элементов образуется из символов и команд исходного файла), то он ничего не знает о том, как они будут выглядеть на экране или листе бумаги. Все задачи по размещению элементов на странице решаются с использованием объектов, которые называются боксы (box).

Бокс — это двумерный объект прямоугольной формы, который измеряется тремя величинами, которые называются ширина, высота и глубина. Геометрически это прямоугольник, длина одной стороны которого равна ширине бокса, а длина другой равна сумме высоты и глубины бокса. Ширина измеряет бокс по горизонтали, а высота и глубина по вертикали. На расстоянии высоты бокса от верхней (соответственно на расстоянии глубины бокса от нижней) стороны прямоугольника проведен отрезок между боковыми сторонами, перпендикулярный им. Этот отрезок называется базовой линией бокса. Точка, находящаяся на пересечении базовой линии и левой стороны прямоугольника, называется точкой привязки.

TEX различает боксы двух типов — простые *символьные* и *линейчатые*. Символьный бокс ассоциирован с символом шрифта, и данные о его измерених хранятся в файле шрифта. *Линейчатые* боксы — это боксы, которые также имеют три измерения, однако они во время вывода будут целиком закрашены. Из таких боксов обычно делаются горизонтальные и вертикальные линейки.

Когда боксы склеиваются горизонтально, точки привязки соседних боксов лежат на одной прямой, при создании вертикальных боксов они также располагаются на одной прямой, только уже вертикальной.

TEX может делать боксы, состоящие из боксов, и работать с ними как с простыми боксами. На этом основан алгоритм создания выходного документа. Сначала символьные и линейчатые боксы склеиваются горизонтально до тех пор, пока не будет набрана строка, которая сама становится боксом. Затем боксы-строки склеиваются вертикально, формируя более крупный бокс, которым является страница документа. В более сложных случаях вложенность боксов может быть и более глубокой.

Для создания горизонтального бокса используется команда `\hbox`, вертикального — `\vbox`.

Для хранения боксов TEX использует боксовые регистры `\box0...\box255`. Устанавливается значение регистра командой `\setbox`.

В отношении того, как TEX работает с боксами, часто использовалось слово «склеивает» для обозначения способа фиксации взаимного расположения боксов по вертикали и горизонтали. На самом деле, соединение боксов между собой происходит посредством объекта, который действительно называется «клей».

Клей представляет собой объект, характеризующийся тремя параметрами: величина, растяжимость и сжимаемость. *Величина клея* — то расстояние по горизонтали или вертикали, которое он будет занимать по умолчанию (образно выражаясь, при отсутствии воздействий факторов внешней среды). *Растяжимость* задает то расстояние, которое клей может занять дополнительно, не ухудшая, с точки зрения типографии, качества взаимного расположения боксов. *Сжимаемость* показывает на какое расстояние может быть изменена величина клея в меньшую сторону. Клей никогда не будет сжиматься больше заданной ему сжимаемости. Однако,

В общем случае параметры клея задаются как

```
<размер1> [plus<размер2>] [minus<размер3>]
```

где <размер1> величина клея, <размер2> его растяжимость, а <размер3> — сжимаемость. Все размеры задаются с обязательным указанием единиц измерения, как было описано ранее. Задание растяжимости и сжимаемости не является обязательным, если они не заданы, то они считаются равными нулю.

Горизонтальный клей задается командой \hskip, вертикальный — командой \vskip. Существует ряд команд, которые задают клей, уже определенный ранее. Например,

```
\smallskip, \medskip, bigskip.
```

Горизонтальный клей с бесконечной растяжимостью устанавливается командой \hfill, вертикальный командой \vfill. Клей, которые имеет как бесконечную растяжимость, так и бесконечную сжимаемость, устанавливается командами \hss и \vss.

Лекция 10: Таблицы в LaTeX

Содержимое таблицы организуется в колонки (графы) и горизонтальные строки таким образом, что каждый элемент является составной частью и строки, и колонки.

Основные элементы таблицы:

- нумерационный и тематический заголовки (номер таблицы и её название);
- головка (заголовочная часть таблицы);
- хвост (вся остальная часть таблицы без головки);
- боковик (первая слева графа таблицы);
- прографка (хвостовая часть таблицы без боковика).

Все данные таблицы содержатся в прографке.

Лекция 10: Таблицы в LaTeX

Для изображения простейших таблиц достаточно LaTeX'овского окружения `tabbing`, которое воспроизводит принцип печати таблицы на печатной машинки. Этот принцип легко просматривается из примера:

```
\begin{tabbing}
МММММММММММ \= ММММММ\= \kill
СИБИРЬ\> Лето \> Зима \\
Жара  \> да   \> нет \\
Мороз \> нет  \> да
\end{tabbing}
```

СИБИРЬ	Лето	Зима
Жара	да	нет
Мороз	нет	да

Первая строка — управляющая. Команда `\=` устанавливает положение табулятора. Команда `\kill` завершает управляющую строку. В результате действия этой команды первая строка не будет напечатана, но все позиции табуляции, установленные в ней, запомнятся LaTeX'ом, и их можно будет использовать в последующих строках.

Лекция 10: Таблицы в LaTeX

Самым популярным окружением для отображения таблиц в LaTeX является окружение `tabular`. Окружение `array` практически полностью идентично `tabular` по своим возможностям. Отличие этих окружений состоит в том, что `array` работает в математической моде, в лекции 5 мы использовали его для создания матриц, которые по своей сути тоже являются таблицами.

В общем случае `tabular` определяется следующим образом:

```
\begin{tabular}[позиционирование таблицы]%
                                {формат столбцов}
                <колонки таблицы>
\end{tabular}
```

Окружению `tabular` необходимо задать обязательный аргумент — преамбулу таблицы.

```
\begin{tabular}{lrc}
<колонки таблицы>
\end{tabular}
```

Преамбула, помещаемая в фигурных скобках непосредственно после `\begin{tabular}`, представляет собой, в простейшем случае, последовательность букв, описывающих структуру колонок таблицы (буква соответствует колонке). Стандартные случаи:

l — колонка выравнивается по левому краю;

r — колонка выравнивается по правому краю;

c — колонка с центрированным текстом.

Между `\begin{tabular}` и закрывающей окружение командой `\end{tabular}` располагается собственно текст таблицы. Данные делятся на ячейки с помощью символа «логическое И» `&`. Переход на следующую строку контролируется стандартной командой переноса строки `\\`. Пробелы в начале или конце «графы» таблицы игнорируются.

Лекция 10: Таблицы в LaTeX

Перечислим опции в LaTeX'овских окружениях tabular и array:

Стандартные опции	
l	выровнять колонку по левому краю
c	отцентровать колонку
r	выровнять колонку по правому краю
p{ширина}	абзац(ы) заданной ширины выровнять по верхним строкам
@{команды}	вместо межколоночного просвета вставить команды
Дополнительные опции	
m{ширина}	В отличие от p{ширина} центрировать по вертикали.
b{width}	В отличие от p{ширина} выровнять по нижним строкам.
>{команды}	(Перед l, r, c, p, m или b) Вставляет команды непосредственно перед содержимым колонки. В качестве команды можно задать шрифт для всей колонки, ненулевой абзацный отступ (к примеру, \setlength{\parindent}{1cm} — значение, добавляемое к высоте каждой клетки таблицы) или открыть формулу ($).
<{команды}	(После l, r, c, p, m или b) Вставляет команды непосредственно после содержимого колонки, например закрывающий формулы $.
|	Вставляет вертикальную черту |, при этом расширяет расстояние между колонками на ширину черты.
!{декл}	Вставляет декл, не ликвидируя просвета между колонками.
Ф	Если иная буква, например, Ф, заранее определена командой \newcolumntype{Ф}{>{$}c<{\$}}, то эта буква в преамбуле равнозначна фрагменту >{$}c<{$}.

Покажем, как в одной и той же таблице можно объединить колонки и столбцы:

```
\begin{tabular}{|*{3}{c|}}
\textbf{ИЛИ} & Истина & Ложь\\[2mm]\hline\hline
Истина & \multicolumn{2}{c|}{Истина}\\
Ложь & Истина & Ложь
\end{tabular}

\begin{tabular}{|*{3}{m{1.2cm}|}}
\textbf{ИЛИ} & Истина & Ложь\\[2mm]\hline\hline
Истина & \multirow{2}{1.2cm}{Истина} & Истина \\
Ложь & & Ложь
\end{tabular}
```

Лекция 10: Таблицы в LaTeX

Иногда для оформления клетки на перекрестии боковика и головк уместно использовать косую черту. В пакете slashbox определена ко манда \backslashbox{текст}{текст}, делящая клетку на две част наклонной чертой. Это может пригодиться, например, для следующего ви да таблиц:

```
\begin{tabular}{|l||*{4}{c|}}\hline
\backslashbox{Помещение}{Дата}
&\makebox[3em]{28/10}&\makebox[3em]{29/10}
&\makebox[3em]{30/10}&\makebox[3em]{31/10}\\ \hline
\hline Малый зал                &&&&\\\hline
       Лекционная аудитория &&&&\\\hline
       Малая аудитория          &&&&\\\hline
\end{tabular}
```

Помещение \ Дата	28/10	29/10	30/10	31/10
Малый зал				
Лекционная аудитория				
Малая аудитория				

Лекция 10: Таблицы в LaTeX

Для раскрашивание таблицы в разные цвета предназначен пакет colortbl (см. документацию colortbl.pdf). Покажем возможности совместного употребления пакетов colortbl и multirow. Таблица:

два \ один	Истина	Ложь
Истина	Истина	Истина
Ложь		Ложь

набрана следующим образом:

```
\begin{tabular}%
{c>{\columncolor{red}}cc}
\backslashbox{два}{один}&Истина&Ложь\\
\rowcolor{red}
Истина & & Истина \\
Ложь & \multirow{-2}*{Истина} & Ложь
\end{tabular}
```

1. Управление формой абзацев

1.1. Управление способом обрывания строки

Режим форматирования абзацев LaTeX позволяет повлиять на режим форматирования целых абзацев. Декларации

```
\sloppy     \fussy
```

изменяют способ обрезания строки. Декларация `\fussy`, действующая по умолчанию, препятствует образованию слишком больших пробелов между словами, но разрешает словам вылезать за правую границу страницы, если не найдена подходящая точка для переноса. Декларация `\sloppy` почти всегда обрывает строку в точности на правом краю, но при этом могут образоваться очень длинные промежутки между словами. В этом случае LaTeX выдаст предупреждение `Underfull \hbox` о незаполненном горизонтальном боксе. Область действия декларации `\sloppy` определяется следующим образом. LaTeX выполняет разбиение на строки, когда доходит до конца абзаца (то есть до пустой строки) или до процедуры, начинающей печать с новой строки. Следовательно, форматирование абзаца осуществляется в соответствии с декларацией, действующей на этот момент. В остальном область действия декларации `\sloppy` определяется как обычно. Она простирается до декларации `\fussy`, возвращающей LaTeX в обычный режим, или же может быть ограничена фигурными скобками.

Абзацы с регулируемыми отступами

Пусть нам потребовалось создать абзац с «отрицательным» абзацным отступом, в котором все строки, кроме первой, начинаются на расстоянии $2\,cm$ от полей. Этого можно добиться следующим образом:

```
\hangindent=2cm \noindent
Здесь \TeX'овский парамет ...
```

Здесь TEX'овский параметр `\hangindent` (по-английски: hanging indentation) означает величину отступа от полей во всех строках абзаца, кроме первой (по умолчанию значение этого параметра равно нулю). Мы начали абзац командой `\noindent`, чтобы первая строка не началась с абзацным отступом.

Пусть теперь нам хочется, чтобы дополнительный отступ, величина которого задана параметром `\hangindent`, начинался не со второй строки, а, скажем, с третьей. Для этого надо установить еще один TEX'овский параметр, обозначаемый `\hangafter`.

Можно также добиться того, чтоб дополнительный отступ не начинался после какой-то строки, а напротив, присутствовал только в нескольких первых строках абзаца.

Для этого надо присвоить параметру `\hangafter` отрицательное значение: если величина `\hangafter` равна $n < 0$, то дополнительный отступ, равный `\hangindent`, будет присутствовать в строках номер $1, 2, \ldots |n|$. С помощью рассмотренных средств TEX'а можно выкапывать в абзацах небольшие ямки. На место образующегося белого прямоугольника можно, например, вклеить иллюстрацию. Данный параграф начинается с команд

```
\hangindent=5cm\hangafter=-3 \noindent.
```

Можно сделать так, что дополнительный отступ, задаваемый параметром `\hangindent`, будет делаться не слева, а справа.

Возможности пакета shapepar

Для того, чтобы верстать абзацы причудливых форм в стандартную комплектацию LaTeX 2_ε входит специальный пакет `shapepar`. Команды пакета придают любому абзацу экзотическую форму из заданного набора. Например, такую форму, как у этого абзаца, можно придать любому другому при помощи команды `\heartpar{текст абзаца}`. О других командах и возможностях пакета можно узнать из спецификации к нему.

♡

```
\begin{alltt}
В процедуре \texttt{alltt} можно
переключать шрифты и набирать
математические формулы. Например:
        \( x\sp{2}+y\sp{2}=Z\sb{0}. \)
\end{alltt}
Пробелы учитываются!
```

В процедуре alltt можно
переключать шрифты и набирать
математические формулы. Например:

$$x^2 + y^2 = Z_0.$$

Пробелы учитываются!

Команды `\(`, `\sp`, `\sb` и `\)`, использованные здесь, чтобы набрать формулу $x^2 + y^2 = Z_0$. Отметим только, что вне процедуры `alltt` ту же формулу можно было бы записать: `$x^{2}+y^{2}=Z_{0}$`, однако внутри `alltt` символы `$`, `^`, `_` не имеют командного значения, которым они обладают в математических формулах.

При помощи \label и \ref можно делать ссылки из вашего текста на плавающий объект.

Следующий пример рисует квадрат и вставляет его в документ. Подобную технику можно использовать, чтобы оставить в документе место под изображения, которые вы вставите позже.

Изображение квадрата, помеченное номером 11, — это место для нового рисунка.

```
\begin{figure}[!hbp]
\makebox[\textwidth]{\framebox[3cm]{\rule{0pt}{3cm}}}
\caption{Три на три сантиметра} \label{white}
\end{figure}
```

Три на три сантиметра

В этом примере LaTeX будет стремиться разместить иллюстрацию прямо по месту (h), а случае неуспеха попытается разместить ее внизу страницы (b). Если не удастся поместить иллюстрацию на текущей странице, он выяснит, можно ли создать страницу плавающих объектов, содержащую эту иллюстрацию и, возможно, некоторые таблицы из очереди таблиц.

При односторонней печати все страницы считаются правыми. По умолчанию для печатного документа класса `article`, `proc`, `report`, `slides` и `letter` устанавливается односторонняя печать, для класса `book` двусторонняя.

Опция `twoside` в декларации `\documentclass` поможет установить режим двусторонней печати также для статей и отчетов, но в письмах, слайдах и научных докладах всегда используется односторонний формат страницы. Информация в колонтитулах, которая чаще всего включает в себя номера страниц, а иногда и названия текущей главы и раздела, призвана помочь читателю найти нужное место в печатном документе.

Стиль страницы (полосы набора), установленный классом печатного документа, изменяют декларации

```
\pagestyle{page-style}     \thispagestyle{page-style}
```

Область действия декларации `\pagestyle` подчиняется обычным правилам с одним уточнением: она начинает действовать с текущей страницы. Декларация `\thispagestyle` аналогична `\pagestyle`, но воздействует только на текущую страницу. Аргумент `page-style` может принимать четыре значения:

- `plain` печатает номера страниц в нижнем колонтитуле, а верхний колонтитул пуст. Стандартные классы, кроме `book`, используют стиль `plain` по умолчанию; empty означает, что верхний и нижний колонтитулы пусты. LaTeX поддерживает нумерацию страниц, но номера не печатает;

- `headings` печатает номера страниц и другую информацию, определяемую классом документа (обычно заголовки разделов), в верхнем колонтитуле. Нижний колонтитул пуст. Печатный документ класса `book` использует стиль `headings` по умолчанию;

- `myheadings` означает, что содержание верхнего колонтитула должен задать сам пользователь, используя команды `\markboth` и `\markright`, описанные ниже. Нижний колонтитул пуст.

Следующая декларация присваивает счётчику страниц page значение 1 и определяет стиль нумерации страниц.

```
\pagenumbering{num-style}
```

Она имеет глобальную область действия. Аргумент num-style должен совпадать с именем одной из команд (за исключением \fnsymbol), которые печатают значения счётчиков, но не иметь обратного слеша:

arabic для печати номеров страниц арабскими цифрами,

roman — строчными римскими цифрами,

Roman — прописными римскими цифрами,

alph — строчными латинскими буквами,

Alph — прописными латинскими буквами,

asbuk — строчными русскими буквами,

Asbuk — прописными русскими буквами.

По умолчанию нумерация страниц ведется арабскими цифрами. Пе-

2. Настройка макета

При размещении изображения на листе предусмотрены поля шириной 1 дюйм, т. е. 2,54 см от верхней и нижней кромки листа. Величину полей можно увеличить или уменьшить, изменив значения командных длин

```
\hoffset=2mm      \voffset=-5mm
```

Значения \hoffset и \voffset, по умолчанию равные нулю, прибавляются к стандартной величине полей, т. е. к 1 дюйму, соответственно, слева и сверху.

Далее приводится набор всех геометрических параметров, управляющих геометрическими параметрами страницы.

\textheight — высота текста, т. е. вертикальный размер тела страницы (без колонтитулов).

\textwidth — ширина текста, т. е. горизонтальный размер колонтитулов и тела страницы.

\oddsidemargin — расстояние между левым краем листа и левым краем текста на правой странице минус 1 дюйм.

\evensidemargin — то же самое для левосторонней страницы. сстояние между верхним краем листа и верхним колонтитулом минус 1 дюйм.

\headheight — высота верхнего колонтитула.

`\headsep` — вертикальное расстояние между верхним колонтитулом и телом текстовой страницы.

`\topskip` — минимальное расстояние между верхним краем тела страницы и нижним краем (т. е. базисной линией) первой строки текста. Соответствует

`\baselineskip` (см. ниже) для первой строки страницы.

`\footheight` — высота нижнего колонтитула.

`\footskip` — вертикальное расстояние между нижним краем текста и нижним краем нижнего колонтитула; измеряется между базисными линиями последних строк текста нижнего колонтитула.

`\linewidth` — ширина строки; равна значению `\textwidth` за исключением строк внутри процедур форматирования абзацев, таких как `quote` или `itemize`. Значение `\linewidth` не должно изменяться командами, изменяющими длину.

`\parindent` — ширина отступа в начале абзаца. Внутри парбокса устанавливается равной нулю. Может быть изменена в любом месте.

3. Печать в две колонки

LaTeX форматирует печатный документ в две колонки, если в списке опций `\documentclass` указать `twocolumn`. Опцию `twocolumn` следует использовать, если необходимо напечатать в две колонки большую часть текста. В одноколоночном документе переход к печати в две колонки осуществляет команда

```
\twocolumn[text].
```

Она начинает новую страницу и начинает печать следующего за ней текста в две колонки. Если имеется необязательный аргумент `text`, он будет напечатан в верхней части новой страницы в одну колонку; если опция отсутствует, то печать в две колонки начинается от начала страницы.

Возврат к одноколоночному формату осуществляет команда

```
\onecolumn[text].
```

Лекция 13: Создание собственных стилевых файлов

На этой лекции Вы узнаете, как и зачем создавать собственные стилевые файлы для системы LaTeX 2_ε.

LaTeX представляет собой такую систему подготовки документов, которая позволяет автору сосредоточится собственно на тексте, не слишком заботясь о том, как отформатировать его. Форматирование достигается посредством применения *стилевых файлов*, в которых содержится информация о том, как логическую структуру документа, созданную с помощью команд LaTeX, привести в вид, пригодный для печати.

В актуальной на сегодняшний день версии LaTeX 2_ε понятие стилевых файлов претерпело некоторые изменения. Они были разделены на два подмножества — на *классы (classes)* и *пакеты (packages)*. Это разделение хоть и является во многом условным (физически, и классы и пакеты содержат одно и тоже — команды LaTeX), однако логически оно достаточно однозначно. А именно, пакеты содержат те особенности форматирования, которые не зависят от используемого класса документа.

Кроме этого, существенное отличие от предыдущей версии LaTeX 2.09 заключается в новом наборе специальных команд для написания пакетов и классов, тогда как раньше создатели стилевых файлов вынуждены были довольствоваться низкоуровневыми командами TeX. LaTeX 2_ε предоставляет в распоряжение всем желающим команды высокого уровня. Более того, стало гораздо проще создавать классы и пакеты, унаследованные от уже существующих.

Поддержка стилевых файлов предыдущих версий

Пакет doc и утилита docstrip

Подробное описание doc и docstrip содержится в файлах doc.dtx и docstrip.dtx дистрибутива LaTeX 2_ε.

Класс или пакет?

Прежде чем поместить несколько новых команд в стилевой файл необходимо решить, что именно это будет: класс или пакет. Правило, которым можно руководствоваться, звучит так: *если команды могут использоваться с любым классом, то это пакет, в противном случае это класс!*

Напомним, что существует два типа классов — основные, такие как article, report или letter, и производные, созданные на базе одного из основных классов.

Именование команд

LaTeX включает три типа команд. Во-первых, авторские команды, такие как

```
\section, \texttt  и   \dots,
```

большинство из них достаточно короткие и все буквы в именах в нижнем регистре. Во-вторых, команды для разработчиков классов и пакетов, с длинными именами, состоящими из букв в разных регистрах, такие как

```
\InputIfFileExists,
\RequirePackage,
\PassOptionsToClass.
```

В третьих, внутренние команды TeX, такие как

```
\@tempcnta, \@ifnextchar, \@eha.
```

Большинство таких команд используют символ @ в имени, т. е. не могут использоваться в документах, только в классах или пакетах. К сожалению, исторически сложилось так, что указанные различия между типами команд достаточно условны. Например, команда \hbox — внутрен-

1.2. Общие рекомендации

Новая система стилевых файлов предоставляет в Ваше распоряжение достаточно команд для того, чтобы сделать хорошо структурированный класс или пакет, который будет эффективным и переносимым.

Подгрузка других файлов

Следующие команды LaTeX предназначены для использования классов и пакетов внутри других классов или пакетов:

```
\LoadClass        \LoadClassWithOptions
\RequirePackage   \RequirePackageWithOptions
```

Настоятельно рекомендуется использовать именно эти команды, а не команду `\input`, по нескольким причинам.

Во-первых, файлы, загруженные при помощи `\input`, не будут отражены в списке `\listfiles`. Во-вторых, если пакет загружается командой `\RequirePackage` (или `\usepackage`), то он будет загружен только один раз, даже если команда встретится несколько раз. В третьих, `\input` не позволяет передать пакету опции при вызове. Таким образом, использование `\input` может быть оправдано только требованиями совместимости с более ранними версиями.

Эффективность

При создании классов и пакетов считается хорошей практикой использовать там, где это возможно, команды LaTeX. То есть, вместо `\def` рекомендуется использовать `\renewcommand` или `\providecommand`, для определения окружения лучше использовать `\newenvironment` или `\renewenvironment` вместо `\def\foo{...}` и `\def\endfoo{...}`. В случае необходимости изменить значения регистров используйте `\setlength`. Для манипуляций с боксами прибегните к помощи команд `\sbox`, `\mbox` и `\parbox`. Пользуйтесь командами `\PackageError`, `\PackageWarning` или `\PackageInfo` вместо `\@latexerr`, `\@warning` или `\wlog`. Не смотря на то, что по-прежнему можно объявлять опции посредством `\ds@<опция>` и вызова `\@options`, рекомендуется использовать для этого команды `\DeclareOption` и `\ProcessOptions`. Они быстрее работают и используют меньше памяти.

2. Структура класса и пакета

Классы и пакеты в LaTeX 2_ε более структурированы чем стилевые файлы предыдущих версий. Они состоят из следующих частей:

- *Идентификация.* В этой части указывается что это класс или пакет LaTeX 2_ε и дается краткое его описание.
- *Предварительные определения.* В этой секции определеются некоторые команды и подгружаются другие файлы. Обычно, эти команды необходимы только для следующей секции описания опций.
- *Опции.* В этой секции объявляются и обрабатываются опции.
- *Дополнительные определения.* В этой части решаются основные задачи: объявляются переменные, определяются команды и шрифты, подгружаются другие файлы и т.д.

2.4. Несколько примеров

Минимальный файл класса

В большинстве новых классов и пакетов определяются новые команды, переопределяются старые или каким-либо образом меняются параметры отображения документа. Суть этих действий целиком зависит от воли автора класса или пакета. Однако, существует три вещи, которые обязан содержать любой класс: определение `\normalsize` и значения для `\textwidth` и `\textheight`. То есть минимальный класс должен выглядеть следующим образом:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{minimal}%
[1995/10/30 Standard LaTeX minimal class]
\renewcommand{\normalsize}%
{\fontsize{10pt}{12pt}\selectfont}
\setlength{\textwidth}{6.5in}
\setlength{\textheight}{8in}
```

Отметим, что такой класс сейчас входит в состав дистрибутива LaTeX и называется `\minimal.cls`. Этот класс не поддерживает множества возможностей, без которых трудно представить себе плодотворную работу ав-

1.1. Принципы работы TEX

Любой класс, пакет или документ TEX представляет собой текстовый файл, включающий в себя, помимо простого текста, последовательности символов специального назначения, обычно начинающиеся с символа обратной косой черты (backslash). Это так называемые «команды» «языка» TEX. В результате обработки входного файла программой TEX, формируется выходной файл формата DVI, который уже не содержит никаких «теховских> команд, а содержит двоичные контрольные последовательности, управляющие процессом вывода текста на экран или печать. Таким образом, TEX является транслятором текстовых файлов определенного формата.

1.2. Трансляция, компиляция и интерпретация

Цель трансляции — преобразовать текст с одного языка на другой, который понятен адресату текста. В случае программ, написанных на языках программирования (Паскаль, C, BASIC и т.д.) адресатом является техническое устройство (процессор) или программа-интерпретатор.

Язык процессоров (машинный код) обычно является низкоуровневым. Транслятор, который преобразует программы в машинный язык, принимаемый и исполняемый непосредственно процессором, называется компилятором. Достоинство компилятора: программа компилируется один раз и при каждом выполнении не требуется дополнительных преобразований. Соответственно, не требуется наличие компилятора на целевой машине, для которой компилируется программа. Недостаток: отдельный этап компиляции замедляет написание и отладку и затрудняет исполнение небольших, несложных или разовых программ.

Трансляция и интерпретация — разные процессы: трансляция занимается переводом программ с одного языка на другой, а интерпретация отвечает за исполнение программ. Однако, поскольку целью трансляции как правило является подготовка программы к интерпретации, то эти процессы обычно рассматриваются вместе. Например, языки программирования часто характеризуются как «компилируемые» или «интерпретируемые», в зависимости от того, преобладает при использовании языка компиляция или интерпретация.

1.3. Формат DVI

Итак, TEXпредставляет собой транслятор с исходного языка (который тоже называется TEX, что иногда вызывает путаницу) на язык, управляющий выводом текста и графики на экран или принтер. Этот язык называется DVI (от англ. DeVice Independent).

Здесь необходимо отметить, что язык программирования TEX является полным по Тьюрингу, т.е. на нем можно реализовать любую вычислимую функцию. Об этом речь пойдет далее.

Файлы DVI не приспособлены для чтения человеком; они содержат двоичные данные, описывающие визуальное представление документа способом, не ориентированным на какой-либо формат изображения, монитор или принтер (отсюда название формата). Файлы DVI обычно подаются на вход другой программы (называемой DVI-драйвером), которая преобразует их в графические данные. Например, многие дистрибутивы TEX включают программу предпросмотра файлов DVI на мониторе — эта программа является драйвером. Драйверы также используются для конвертирования файлов DVI в другие форматы (такие как PostScript, PDF) и для печати. О формате (или, правильнее, о языке) PostScript мы поговорим более подробно.

2. Язык PostScript

PostScript (Постскрипт) — язык описания страниц, в основном используемый в настольных издательских системах.

Postscript был разработан Джоном Уорноком и Чаком Гешке из Adobe Systems в начале 80-х гг. Исходно Postscript использовался как ядро механизма печати компьютеров Apple, но вскоре стал широко распространенным стандартом для большинства компьютерных систем. Интерпретаторы Postscript (в виде программных или аппаратных компонентов) для печати документов присутствуют практически во всех современных компьютерных системах. В Postscript используется модель изображения текста (или рисунков) на чистой странице. Когда страница готова, она выводится на печать и начинается «прорисовка» изображения очередной страницы.

В 1984 году увидел свет первый вариант PostScript (позже к названию было добавлено Level 1, чтобы отличать его от Level 2). Postscript имел огромные преимущества перед другими системами того времени:

- Платформонезависимость. Один и тот же файл мог печататься как на лазерном принтере, выдававшем тогда 300 dpi, так и на фотонаборном устройстве с 2400 dpi с наилучшим качеством в каждом случае.
- Любой производитель мог лицензировать интерпретатор PostScript и использовать PostScript со своим устройством.
- Спецификации PostScript были общедоступны, таким образом, любой разработчик мог писать программы, поддерживающие PostScript.

В 1991 Adobe выпустила следующую версию PostScript — PostScript Level 2. Основными улучшениями были:

2.5. Пример программы

```
%!PS-Adobe-2.0
/Times-Italic findfont % установить шрифт Times курсив
30 scalefont           % размером 30 пунктов
setfont

/printZip {                    % процедура вывода текста
0 0 moveto                     % перейдем в точку 0 0
(Can you see a shadow ?) show% выведем на экран текст
} def                          % конец описания процедуры
30 500 translate  % установить начало координат
```

```
.95 -.05 0 {      % цикл от 0,95 до 0 с шагом -0,05
setgray           % установить цвет в оттенок серого,
             %равный текущему значению параметра цикла
printZip          % вызвать процедуру вывода на экран
-.5 .5 translate % сместить начало координат
} for             % конец цикла

1 setgray printZip% вывести текст белым цветом

showpage           % вывести страницу
```

Если сохранить приведенный выше текст в файле с расширением `.ps` и просмотреть или вывести на печать, то должна получиться примерно такая картинка:

Рис. 14.1. Результат работы программы

Как уже упоминалось выше, TEX (как язык программирования) является функционально полным, т.е. на нем можно реализовать любую вычислимую функцию. Среди всех возможностей языка для создания пригодных для печати документов используется лишь малая часть, связанная непосредственно с набором. Это определение и переопределение команд, счетчиков, размеров и пр. Это вполне оправдано, поскольку позволяет автору, пользующемуся TEX, не быть программистом. Да и у программистов обычно не возникает желания использовать TEX для разработки, например, информационных систем. Однако, для создания собственного формата или сложного стиля и для решения других технических задач знание этих возможностей не будет лишним.

1. Определения и переопределения в LATEX

Формат LATEX отличается от многих других форматов TEX тем, что в нем реализована идея, высказывавшаяся Д. Кнутом в его книге «The TEXbook», о создании системы, основанной на TEX, однако замкнутой в отношении множества команд, т.е. такой, что любая ее функция реализуется только посредством ее собственных команд, не используя примитивы TEX. Соответственно, для определения и переопределения объектов LATEX также использует собственные команды, о которых и пойдет дальше речь.

1.1. Команды

Для определения команд используется конструкция

```
\newcommand{<cmd>}[<args>]{<definition>}
\newcommand{<cmd>}[<args>][<default>]{<definition>}
```

<cmd> определяет имя новой команды, и должно начинаться с символа \. Количество аргументов команды поределяется целым числом <args>. Максимальное значение равно 9. По умолчанию команда определяется без аргументов. Значение аргументов по умолчанию задается необязательным параметром <default>. Определение <definition> содержит текст, которым будет заменяться встреченная в документе команда \cmd. В определении команды параметры в количестве, заданном <args>, обозначаются символом # с идущим за ним номером аргумента, от 1 до 9 соответственно. Если команда \cmd уже определена, будет выдано сообщение об ошибке. Для переопределения уже существующих команд используются

```
\renewcommand{<cmd>}[<args>]{<definition>}
\renewcommand{<cmd>}[<args>][<default>]{<definition>}
```

соответственно, которые имеют точно такой же синстаксис.

1.2. Счетчики

Команда

```
\newcounter{<name>}[<counter>]
```

создает счетчик с именем name и со значением по умолчанию равным 0. Необязательный параметр <counter> позволяет сослаться на другой существующий счетчик с именем counter. В этом случае значение нового счетчика будет сбрасываться в 0 при увеличении значения counter.

1.3. Окружения

Окружения позволяют ограничивать часть текста, который должен быть отформатирован способом, существенно отличающимся от

```
\newenvironment{<name>}[<args>]{<begdef>}{<enddef>}
\newenvironment{<name>}[<args>][<default>]%
                    {<begdef>}{<enddef>}
\renewenvironment{<name>}[<args>]{<begdef>}{<enddef>}
```

<name> указывает на название нового либо переопределяемого окружения, целое число <args> определяет количество аргументов у окружения (по умолчанию нет аргументов). Текст <begdef> заменяет появившуюся в документе инструкцию \begin{name}, внутри можно использовать имена параметров, которые обозначаются парами #<номер параметра>, например #1. <enddef> подставляется соответственно вместо \end{name}, использовать здесь параметры нельзя.

1.4. Шрифты

Для определения новых имен шрифтов используется команда

```
\newfont{<cmd>}{<font_name>}
```

<cmd> — имя новой команды, которая ранее не должна быть определена. <font_name> — наименование шрифта, который будет выбран при появлении в документе команды \cmd.

1.5. Размеры

```
\newlength{\<name>}
```

определяет новый размер с именем <name> и значением 0 по умолчанию. Если команда с таким именем уже определена, то возникнет сообщение об ошибке.

2. Макросы TEX

Макрос (или макроподстановка) позволяет заменять одну или несколько команд одной командой. Приведенные выше конструкции решают подобные задачи, однако они сами являются макросами, написанными с использованием примитивов TEX. Далее мы будем рассматривать возможности, предоставляемые базовой системой TEX.

2.1. Упрощение ввода длинных последовательностей команд

Предположим, что в тексте часто встречается какая-то последовательность инструкций, например это может быть обозначение точки n-мерного пространства $(x_1, \ldots, x_n)$, которое в файле выглядит как `(x_1, \dots, x_n)`. TEX

```
\def\<name>[<params>]{<definition>}
```

Название новой команды задается аргументом `<name>`. Необязательный аргумент `<params>` задает количество и формат входных параметров новой команды, которые обозначаются символом `#` и цифрой от 1 до 9. Определение новой команды `<definition>` содержит текст, который TEX будет подставлять всегда, когда встретит команду `\<name>`. Внутри `<definition>` можно использовать параметры, перечисленные в `<params>`. Причем если в `<params>` параметры должны перечисляться обязательно по порядку, то в `<definition>` они могут использоваться несколько раз и в произвольном порядке.

Лекция 15: Создание новых команд. Программирование на языке TeX

Приведем пример из книги Д. Кнута «The TEXbook», описывающий макрос \primes{n}, который раскрывается в список первых n простых чисел.

```
\newif\ifprime\newif\ifunknown % булевские переменные
\newcount\n\newcount\p\newcount\d \newcount\a  %целые
\def\primes#1{2,~3
              % предполагаем, что #1 как минимум равен 3
\n=#1 \advance\n by-2                % количество чисел n
\p=5                            % первое простое больше 3
\loop\ifnum\n>0 \printifprime\advance\p by2 \repeat}
\def\printp{,              % макрос печати простого числа
  \ifnum\n=1 и~\fi %перед последним числом печатаем 'и'
  \number\p \advance\n by -1 }
\def\printifprime{\testprimality \ifprime\printp\fi}
\def\testprimality{{\d=3 \global\primetrue
\loop\trialdivision \ifunknown\advance\d by2 \repeat}}
\def\trialdivision{\a=\p \divide\a by\d
 \ifnum\a>\d \unknowntrue\else\unknownfalse\fi
  \multiply\a by\d
  \ifnum\a=\p \global\primefalse\unknownfalse\fi}
```

1. Пакет color

Поддержка цвета в LaTeX обеспечивается пакетом `color`. Цветные таблицы позволяет создавать пакет `colortbl`. Далее рассматриваются эти два пакета.

1.1. Драйвер цветной печати

Упомянутые пакеты основаны на возможностях, которые не относятся непосредственно к TeX. Эти возможности поддерживаются так называемым драйвером, продназначенным для печати файла `dvi`. К сожалению, не все драйверы поддерживают одинаковые возможности, так что графические пакеты обычно имеют опции, в которых указывается какой драйвер нужно использовать. Перед использованием пакета рекомендуется задать драйвер по умолчанию. Сделать это можно в конфигурационном файле с расширением `cfg`. Например, для пакета `color` создается файл `color.cfg`, в первой строке которого указывается

```
\ExecuteOptions{dvipsone}
```

Теперь драйвером по умолчанию будет `dvipsone`. Перечень доступных для использованиея драйверов выглядит так:

```
dvips,xdvi,dvipdf,dvipdfm,dvipdfmx,pdftex,dvipsone,
dviwindo,emtex,dviwin,pctexps,pctexwin,pctexhp,
pctex32,truetex,tcidvi,vtex,oztex,textures,xetex
```

1.4. Определение цветов

Основные цвета определены по умолчанию, но список цветов можно расширять, пользуясь командой

```
\definecolor{}{<model>}{<colour specification>}
```

<name> это название нового цвета, <model> — используемая для задания цвета модель, и <colour specification> спецификация цвета в выбранной модели. Например,

```
\definecolor{light-blue}{rgb}{0.8,0.85,1}
\definecolor{mygrey}{gray}{0.75}
```

1.5. Использование цвета

Для смены цвета текста используется команда

```
\color{<name>}
```

Она изменяет текущий цвет на <name> до конца текущей группы или окружения. Другая команда

```
\textcolor{<name>}{<text>}
```

принимает текст, цвет каторого должен быть изменен, в качестве аргумента <text>. В некоторых случаях такой синтаксис удобнее, хотя по сути это эквивалентно конструкции {\color{<name>}text}.

1.6. Цвет страницы

Фоновый цвет страницы можно установить командой

```
\pagecolor{<name>}
\pagecolor[<model>]{<specification>}
```

Значения параметров этих команд то же что и у команды \color. Они управляют цветом фона текущей страницы и всех последующих. Вернуть цвет фона в исходное состояние можно только явно, испульзуя команду \pagecolor{white}.

1.7. Цвет фона боксов

Две команды, похожие на \fbox, позволяют делать боксы, залитые выбранным цветом.

```
\colorbox{<name>}{<text>}
\colorbox[<model>]{<specification>}{<text>}
\fcolorbox{<name1>}{<name2>}{<text>}
\fcolorbox[<model>]{<specification1>}%
            {<specification2>}{<text>}
```

\colorbox создает бокс, окрашенный в цвет <name>, либо заданный спецификацией в определенной модели. Версия \fcolorbox создает бокс, залитый цветом <name2> с видимой рамкой вокруг, цвета <name1>. Эти команды используют параметры \fboxrule и \fboxsep для определения толщины линии и размера закрашиваемой области.

2.1. Команда columncolor

Реализация команды в пакете такова, что она может использоваться только как аргумент спецификатора столбца > в преамбуле array или tabular, а также в \multicolumn. Формат команды следующий:

```
\columncolor[<color model>]{<colour>}%
    [<left overhang>][<right overhang>]
```

Аргументы <color model> и <colour> задают цвет подобно тому, как это происходит в команде \color, описанной выше. Аргумент <left overhang> и <right overhang> задают длину, на которую панель выступает за граница содержимого ячейки слева и справа соответственно. Если опущен один из размеров, то второй используется с обеих сторон. Если отсутствуют оба, то используется значение параметров \tabcolsep или \arraycolsep, в зависимости от окружения.

2.3. Команда cellcolor

Можно задать цвет всей таблицы, или ее строк, но иногда бывает необходимо указывать цвет отдельной ячейки. Для этого служит команда

```
\cellcolor[<color model>]{<colour>}%
 [<left overhang>][<right overhang>]
```

которая имеет тот же синтаксис, и цвет, который она задает, преобладает над остальными установками. Использовать ее можно в любом месте ячейки, к которой она применяется.

2.4. Цвет границ ячеек

Можно задать цвет вертикальной границы, не используя никаких дополнительных команд, заменяя обычный разделитель столбцов `|` на `!{\color{green}\vline}`. Однако, раскрасить `\hline` и `\cline` не так просто, так что в пакет предоставляет в распоряжение авторов дополнительные команды.

```
\arrayrulecolor[<color model>]{<colour>}
```

имеет одинаковые параметры с `\color`, и задает цвет всех последующих горизонтальных и вертикальных границ во всех таблицах документа.

Следующего результата раскраски границ таблицы

содержимое 1.1	содержимое 1.2
содержимое 2.1	содержимое 2.2

можно добиться таким способом:

```
\setlength\arrayrulewidth{2pt}
\arrayrulecolor{blue}
\setlength\doublerulesep{2pt}
\doblerulesepcolor{yellow}
\begin{tabular}{||l||c||}
\hline\hline
содержимое 1.1&содержимое 1.2\\
содержимое 2.1&содержимое 2.2\\
\hline\hline
\end{tabular}
```

Эта лекция посвящена использованию в тексте дополнительных выразительных средств, таких как графические иллюстрации или таблицы. В терминах TEX они обозначаются как плавающие (floating) объекты.

1. Пакет floatflt

Пакет floatflt определяет два окружения: floatingfigure и floatingtable. Основная идея состоит в том, что если рисунок или таблица не очень широкие, то возле них может располагаться текст, справа или слева. Подключается пакет стандартным для LATEX образом

```
\usepackage[<option>]{<floatflt>}
```

где <option> может принимать одно из трех значений: rflt, lflt, или vflt. Эти опции устанавливает позицию для плавающего объекта по умолчанию:

- **rflt** — справа;
- **lflt** — слева;
- **vflt** — справа на нечетных и слева на четных страницах.

Последняя опция выбрана по умолчанию.

1.1. Окружение floatingfigure

Общий формат вызова окружения приведен ниже. Аргумент <option> может принимать значение r, l, p, или v, а <width> задает размер плавающего объекта.

```
\begin{floatingfigure}[<option>]{<width>}
вставка графического объекта
\caption %% если необходимо
\end{floatingfigure}
```

Опции обозначают следующее:

- **r** Объект будет выводиться справа.
- **l** Объект будет выводиться слева.
- **p** Объект будет выводиться справа на нечетных и слева на четных страницах.
- **r** Объект будет выводиться так, как определено опциями пакета, если же они не указаны, то справа на нечетных и слева на четных страницах.

2. Пакет wrapfig

2.1. Окружение wrapfigure

Окружение имеет следующий формат вызова:

```
\begin{wrapfigure}[<number>] {<placement>}%
 [<overhang>] {<width>}
                        <figure>
\end{wrapfigure}
```

Параметр <number> задает количество строк текста для обтекания рисунка. Если параметр отсутствует, то количество определяется автоматически, в зависимости от высоты рисунка.

Параметр <placement> является обязательным, он определяет расположение рисунка на странице. Расположение может задаваться строчными R, L, I, O либо прописными r, l, i, o буквами. Строчные буквы задают желаемое расположение, т.е. предполагают возможность перемещения фигуры при необходимости. Прописные буквы указывают что фигура должна располагаться точно там где указано. Варианты расположения такие:

- **r, R** справа от текста
- **l, L** слева от текста
- **i, I** внутренний край страницы
- **o, O** внешний край страницы

Указать можно только одну из них.

3. Класс NCC

Класс NCC, разработанный А.И. Роженко, представляет собой масштабную доработку стандартных классов и пакетов LaTeX, и заслуживает отдельного обсуждения.

Команды, имеющие самостоятельное значение, вынесены в отдельные пакеты коллекции `ncctools`. Собственно макрос NCC и сопутствующие ему пакеты образуют коллекцию `ncclatex`. Ядро макроса интегрировано пакетом `ncclatex`, который можно использовать с другими классами.

Класс содержит множество полезных команд, которые существенно расширяют стандартные возможности LaTeX. Все они подробно описаны в документации http://www.ctan.org/get/macros/latex/contrib/ncclatex/ncclatex.pdf, том числе и на русском языке http://www.ctan.org/get/macros/latex/contrib/ncclatex/nccnews.pdf.

В частности, отдельно описаны команды, позволяющие оформлять с обтеканием или в виде плавающих объектов рисунки и таблицы.

3.1. Основные команды

Следующая команда задает стиль <style> для оформления плавающих объектов (иллюстраций и таблиц):

```
\FloatingStyle{<style>}
```

По умолчанию заден стиль

```
\FloatingStyle{\footnotesize\centering}
```

Для размещения рисунков и таблиц в виде мини-сираницы служат команды

```
\minifig[<pos>][<высота>][<innerpos>]{<ширина>}{<тело>}
\minitabl[<pos>][<высота>][]{<ширина>}{<тело>}
```

Параметр <pos> определяет как объект будет выравниваться по вертикали. Он может принимать одно из трех значений:

- **t** выравнивание по верхнему краю
- **t** выравнивание по нижнему краю
- **c** выравнивание по центру

3.2. Обтекание текстом

Пакет NCC отличается от описанных выше тем, что в нем реализованы средства ручного контроля за обтеканием. Для размещения объекта в тексте с обтеканием используется одна из команды:

```
\sidefig[<pos>](<w1>)(<w2>){<рисунок>}{<текст>}
\sidefig*[<pos>](<w1>)(<w2>){<рисунок>}{<текст>}
\sidetabl[<pos>](<w1>)(<w2>){<таблица>}{<текст>}
\sidetabl*[<pos>](<w1>)(<w2>){<таблица>}{<текст>}
```

По умолчанию рисунок или таблица располагается на четных страницах слева и на нечетных справа. Версии команд со звездочкой размещают объект на противоположной стороне. Необязательные параметры имеют следующие значения:

- **pos** определяет вертикальное выравнивание относительно текста (t, b или c)
- **w1** задает ширину плавающего объекта в единицах `\unitlength`
- **w2** задает ширину текстового бокса.

3.3. Оформление рисунка или таблицы

Следующие команды фактически являются надстройками над окружениями figure и table:

```
\fig[<placement>](<w>){<рисунок>}
\fig*[<placement>](<w>){<рисунок>}
\tabl[<placement>](<w>){<таблица>}
\tabl*[<placement>](<w>){<таблица>}
```

<placement> задает предпочтительные варианты размещения на странице, <w> задет ширину.

Следующие команды размещают рядом пару плавающих объектов:

```
\figs[<placement>](<w1>)(<w2>)%
                        {<рисунок 1>}{<рисунок 2>}
\figs*[<placement>](<w1>)(<w2>)%
                        {<рисунок 1>}{<рисунок 2>}
\tabls[<placement>](<w1>)(<w2>)%
                        {<таблица 1>}{<таблица 2>}
\tabls*[<placement>](<w1>)(<w2>)%
                        {<таблица 1>}{<таблица 2>}
```

Параметр <placement> имеет то же значение, что и для предыдущих команд, параметры <w1> и <w2> задают ширину левого и правого элемента соответственно.

Главная задача проекта PDFTEX состоит в том, чтобы

- из исходного файла TEX можно было создавать документ в формате PDF;
- расширить возможности TEX за счет использования возможностей формата PDF, например создавать активные гиперссылки.

PDFTEX входит в состав большинства современных дистрибутивов TEX, в ином случае он может быть установлен дополнительно. Получить его можно по адресу ftp://ftp.cstug.cz/pub/tex/local/cstug/thanh/pdftex/latest. Когда все установлено, использовать новые возможности можно просто вызывая для обработки исходного TEX-файла программу pdftex вместо tex (или pdflatex вместо latex), например

```
# pdftex myfile.tex
```

Заметим, что если отключить вывод в формат PDF, то PDFTEX создаст файл в привычном формате DVI.

Все возможности PDFTEX подробно описаны в документации. Далее мы познакомимся с некоторыми из них.

1. Информация о документе

Просматривая документ в формате PDF программой Adobe Acrobat Reader, получить информацию о документе можно нажав сочетание клавиш Ctrl–D (или выбрав в меню Файл пункт Свойства документа). Добавить такую информацию в свой файл PDF можно командой

```
\pdfinfo{<info>}
```

где <info> задается в стиле PDF, в виде пар

```
/fieldname (<text>)
```

Возможные значения для /fieldname такие:

```
/Title
/Author
/Creator
/Producer
/CreationDate
/ModDate
/Subject
/Keywords
```

Дата задается в формате `D:YYYYMMDDHHmmss`. Пакет `datetime` (начиная с версии 2.31) определяет команду `\pdfdate`, которая вставляет дату в правильном формате. Приведем пример:

```
\pdfinfo{
/Author (The Author)
/Title (Creating PDF using PDFTeX)
/CreationDate (D:200812312359)
/ModDate (D:\pdfdate)
/Subject (PDFTeX)
/Keywords (PDF;TeX)
}
```

Используя команды PDFTEX необходимо помнить, что для обычного TEX (или LATEX) эти команды неизвестны, и вызовут ошибку. Для того, чтобы сделать исходный файл универсальным, необходимо иметь возможность проверить, какой программой он обрабатывается. Это можно сделать, подключив пакет `ifpdf`, в котором определена команда-условие `\ifpdf`, позволяющая проверить, используется PDFTEX или нет. Например, команду `\pdfinfo` рекомендуется использовать следующим образом:

```
\ifpdf
\pdfinfo{
/Author (The Author)
/Title (Creating PDF using PDFTeX)
/CreationDate (D:200812312359)
/ModDate (D:\pdfdate)
/Subject (PDFTeX)
/Keywords (PDF;TeX)
}
\fi
```

2. Включение графики в PDF-документы

Для включения в документы PDF графики можно использовать пакеты graphics (graphicx), которые будут описаны позднее. Здесь только отметим, что, к сожалению, придется отказаться от использования графических файлов в формате PS или EPS, поскольку PDFTEX не умеет преобразовывать такие форматы. Ему нужны либо файлы в формате PDF, либо в одном из поддерживаемых графических форматов, например PNG или TIFF.

3. Гиперссылки. Пакет hyperref

Включение гиперссылок в PDF-документы осуществляется при помощи пакета hyperref. Для детального знакомства с этим пакетом необходимл воспользоваться документацией к нему. Здесь будут приведены только основные команды и примеры их использования.

3.1. Перекрестные ссылки и цитаты

Как только пакет hyperref будет подключен командой \usepackage, все перекресные ссылки и цитаты, созданные командами \ref, \pageref и \cite, автоматически будут сделаны активными гиперссылками. По умолчанию в документе они выглядят как цветные (красные) рамки вокруг текста ссылки. Это поведение может быть изменено одной из опций пакета. Например, если указана опция colorlinks, то активные ссылки будут просто выделяться цветом.

Пакет hyperref определяет также версии команд \ref* и \pageref*, которые печатают соответствующий номер, однако не создают гиперссылки, что позволяет их использовать с другими командами.

3.2. Содержание и закладки

Содержание, производящееся командой `ntableofcontents`, автоматически будет состоять из активных ссылок на соответствующие разделы. По умолчанию ссылками будут сделаны заголовки разделов, а не номера страниц, однако обратить ситуацию можно использовав опцию `linktocpage` пакета `hyperref`.

Если установлена опция `bookmarks`, то будет создан набор закладок документа PDF, доступ к которым может быть получен выбором опции `Закладки` в меню `Окно` программы `Acrobat Reader`. Закладки являются удобным средством навигации по документу. Если также будет использована опция `bookmarksopen`, то по умолчанию дерево закладок будет полностью развернуто. Опция `bookmarksnumbered` добавит к тексту закладки номер раздела, на который она ссылается. Необходимо помнить, что в закладках нельзя использовать никакое форматирование LaTeX, в том числе и математический режим.

3.3. Разметка страницы

Размер страницы устанавливается одной из опций

```
a4paper, a5paper, b5paper,
letterpaper, legalpaper, executivepaper.
```

1. Пакеты graphics и grapicx

В дистрибутиве LaTeX поставляются два пакета для работы с графикой: стандартный пакет для работы с графикой `graphics` и расширенная версия этого пакета `graphicx`. Отличаются они только форматом необязательных параметров команд, обязательные параметры одинаковые для обоих пакетов.

1.1. Опции пакетов

Графические пакеты, как и пакеты управления цветом, зависят от используемого графического драйвера. Использовать разные драйверы в одном документе невозможно, поэтому выбрав драйвер в однгом из пакетов указывать его в другом уже не нужно. Глобальные установки для графических пакетов хранятся в файле `graphics.cfg`. Чтобы использовать драйвер `dvips`, укажите в первой строке этого файла

```
\ExecuteOptions{dvips}
```

Кроме опции драйвера, графические пакеты имеют несколько опций, которые управляют набором предоставляемых возможностей.

- **draft** минимизирует набор используемых специальных возможностей. В частности, рисунки не отображаются, вместо них показывается рамка, соответствующая размеру рисунка и внутри нее текст с названием графического файла с рисунком.
- **final** противоположна `draft`. Можно использовать для отмены действия глобальной опции `draft`, указанной при загрузке класса.
- **hiderotate** Отменяет показ повернутого текста.
- **hidescale** Отменяет показ масштабированного текста.
- **hiresbb** Указывает брать размеры из строки `%%HiResBoundingBox`, а не из стандартного `%%BoundingBox`.
- **demo** команда `\includegraphics` вместо включения графического файла рисует прямоугольник размерами 150 на 100 пунктов, если только другие размеры не заданы вручную.

1.2. Вращения

```
graphics: \rotatebox{<angle>}{<text>}
graphicx: \rotatebox[<key val list>]{<angle>}{<text>}
```

Эти команды создают бокс с текстом `<text>` и поворачивают его на `<angle>` градусов. Стандартная версия всегда поворачивает бокс вокруг опорной точки, а расширенная позволяет изменять это поведение посредством параметров `<key val list>`, которые могут быть следующие

```
origin=<label> x=<dimen> y=<dimen> units=<number>
```

Так что можно задать координаты центра вращения `x` и `y`, либо указать одну из предопределенных точек, задав соответствующий `origin`. Например, `[origin = c]` задает вращение вокруг центра бокса, а `[origin = tr]` — вокруг верхнего правого угла. Параметр `units` позволяет изменить единицы измерения угла вращения, например `[units = -360]` задает вращение против часовой стрелки, а `[units= 6.283185]` позволяет указывать углы в радианах.

1.3. Масштабирование

Графические пакеты поддерживают различные способы масштабирования текста.

```
\scalebox{<h-scale>}[<v-scale>]{<text>}
```

задает масштабные множители по горизонтали (<h-scale>) и вертикали (<v-scale>). Если на задан параметр <v-scale>, то для растяжения по вертикали используется <h-scale>. Команда

```
\reflectbox{<text>}
```

является аббревиатурой (макроопределением) для
\scalebox{-1}[1]{<text>}.

Команда

```
\resizebox*{<h-length>}{<v-length>}{<text>}
```

масштабирует <text> таким образом, чтобы он стал шириной <h-length>. Если в качестве одного из инмерений указан !, то команда будет подбирать по этому измерению такой множитель, чтобы масштабирование по всем направлениям было одинаковым. Обычно размер <v-length> относится именно к высоте бокса, но в версии этой команды со звездочкой он обозначает высоту+глубину бокса.

1.4. Вставка графических файлов

Графические файлы включаются в документ при помощи двух команд: graphics и graphicx.

```
graphics:
    \includegraphics*[<llx,lly>][<urx,ury>]{<file>}
graphicx:\includegraphics*[<key val list>]{<file>}
```

Если в команде присутствует символ *, то рисунок будет обрезан до указанного в необязательном аргументе размера, в противном случае, часть рисунка, выходящая за пределы указанных размеров, будет накладываться на текст. Если необязательные аргументы, задающие размер, отсутствуют, то информация о размере берется из самого графического файла в соответствии с перечисленными ниже правилами.

- **graphics**ли параметр [<urx,ury>] присутствует, то он указывает на координаты правого верхнего угла рисунка, в терминах размеров TEX. Если не указаны единицы измерения, то по умолчанию используется bp. Если присутствует параметр [<llx,lly>], то он указывает координаты левого нижнего угла рисунка, иначе эти координаты принимаются равными [0,0].

- **graphicx**рсия команды со звездочкой в этом пакете оставлена только для обратной совместимости. Она просто добавляет ключ clip к списку ключей <key val list>.

2. Ряд команд пакета graphics

```
\graphicspath{<dir-list>}
```

Эта команда используется для задания списка каталогов для поиска в них графических файлов. Каждая директория в списке должна быть окружена парой фигурных скобок, даже если она в списке одна. Например,

```
\graphicspath{{eps/}{tiff/}}
```

дает указание искать графические файлы в поддиректориях `eps` и `tiff` текущей директории. По умолчанию, TEX будет искать графические файлы там же, где он ищет и остальные. То есть в директориях, заданных

```
\DeclareGraphicsExtensions{<ext-list>}
```

Эта команда определяет поведение TEX в случае, когда не указано расширение графического файла. <ext-list> содержит разделенные запятыми расширения.

Лекция 20: Гипертекст в PDF

Пакет hyperref расширяет функциональные возможности LaTeX'а, делая перекрестные ссылки на команды оглавления, библиографии и т.д. Он также обеспечивает новые команды для ссылок к внешним документам.

Загружаем пакет hyperref:

```
\usepackage[<opt>]{hyperref}
```

При загрузке этого пакета после компиляции документа на выходе по умолчанию появится закладка рядом с текстом (слева), на которой выведется список разделов документа. Если закладка будет не нужна, её можно убрать, написав опцию bookmarks=false (о которой говорится дальше) при загрузке пакета hyperref.

Главы в содержании будут обведены рамочкой красного цвета (цвет шрифта, математики и др. не изменится), а также появится ссылка из содержания к странице, на которой начинается соответствующий раздел документа. Все ссылки внутри документа будут обведены красной рамочкой, ссылки на другой файл обведены темно-зеленой рамочкой, ссылки на сайт (интернет) — голубой рамочкой, ссылки на библиографию — салатовой рамочкой. При нажатии на ссылку, она станет в боксе черного цвета, сама ссылка станет желтого цвета.

Включая опции <opt>, описанные ниже, можно изменять цвета ссылок и внешний вид документа.

1. Команды, определенные в пакете hyperref

Пакет hyperref предназначен в основном для гипертекстовых ссылок. Три описанные ниже команды устанавливают эти ссылки.

```
\href{file}{text}
```

Ссылка на файл file.pdf (указывается имя file файла file.pdf без расширения). С помощью этой команды можно делать ссылку и на сайт в интернете. Для этого вместо file нужно написать адрес сайта в интернете, например, написав

```
\href{http://www.sfu-kras.ru}{ссылку на сайт СФУ}
```

получим: ссылку на сайт СФУ.

```
\hypertarget{name}{text}
```

Эта команда подобна команде \label из LaTeX'а, она ставит видимую метку text. В text можно написать любое слово, знак или символ, а можно вообще ничего не писать (оставить пустые скобки).

```
\hyperlink{name}{text}
```

2. Опции пакета

Пакет hyperref имеет большое количество опций. Опции перечисляются через запятую в необязательном аргументе команды \usepackage при подключении пакета

```
\usepackage[список опций]{hyperref}
```

либо после подключения пакета в аргументе команды

```
\hypersetup{список опций}.
```

Команда \hypersetup используется не только в преамбуле, но и в окружении document для того, чтобы менять различные PDF настройки прямо по ходу документа. Каждая опция в списке задается в формате

```
опция=значение
```

Для опций булевого типа значение true можно опускать.

Мы рассмотрим только основные опции. Более полный список можно найти в документации к пакету.

2.1. Внешний вид гиперссылок

Оформление текста гиперссылки регулирует опции булевого типа:

```
colorlinks,     frenchlinks
```

По умолчанию значения обеих опций равно `false` и текст гиперссылки печатается обычным шрифтом без выделения цветом, но вокруг него рисуется рамка. Если `colorlinks` присвоить значение `true`, то текст гиперссылки будет выделяться цветом, а рамки не будет.

Если же `frenchlinks` присвоить значение `true`, то текст гиперссылок будет печататься шрифтом КАПИТЕЛЬ без выделения цветом. При вычислении координат углов прямоугольника, который становится активной областью гиперссылки на PDF странице, LaTeX добавляет поля, величина которых задается опцией

```
pdflinkmargin=длина
```

Вместе с величиной указываются, как это принято в LaTeX'е, и единицы изменения, например, 10pt. По умолчанию действует значение 1pt. Опцию имеет смысл использовать для увеличения расстояния между рамкой и текстом ссылки.

2.2. Цвет текста гиперссылок

Следующие опции определяют цвет текста ссылок разного типа:

```
linkcolor pagecolor filecolor  citecolor  urlcolor
```

Цвет текста ссылок на мишени внутри документа определяет `linkcolor`; по умолчанию имеет значение red. Цвет текста ссылок, когда мишенью внутри документа является страница, определяет `pagecolor`; по умолчанию имеет значение red. Цвет текста ссылок, которые открывают локальные PDF файлы задает `filecolor`; по умолчанию имеет значение cyan. Цвет библиографических ссылок, которые печатает команда `\cite`, определяет `citecolor`; по умолчанию имеет значение green. Цвет текста ссылок на ресурсы с URL определяет `urlcolor`; по умолчанию имеет значение magenta. Допустимыми значениями перечисленных выше опций могут быть имена цветов из пакета `color` или имена, определенные посредством команды `\definecolor` из того же пакета, причем команда становится доступной только после подключения самого пакета. Опция

```
anchorcolor=цвет
```

задает цвет текста мишени; по умолчанию **black**.

2.3. Окантовка гиперссылок

По умолчанию вокруг гиперссылок рисуется прямоугольная рамка. Линия рамки имеет толщину 1 bp. Опция

```
pdfborder={стиль рамки}
```

позволяет изменить вид рамки. Стиль рамки в общем случае состоит из трех целых чисел и одного массива чисел. Первые два числа — радиусы закругления углов рамки, а третье число - толщина линии рамки. Если в качестве толщины линии указать 0, то рамки не будет. Необязательный массив позволяет заменить сплошную линию на штриховую. Первое число в массиве задает длину штриха, а второе — расстояние между штрихами (не должны превышать 10). Пример: `pdfborder={0 0 2 [6 4]}`. Если расстояние между штрихами совпадает с длиной штриха, то вместо [n n] можно указывать [n]. По умолчанию стиль рамки задан как 0 0 1.

2.4. Выделение активных гиперссылок

```
pdfhighlight=ключ
```

— опция устанавливает выделение гиперссылок, когда кнопка мыши нажимается и держится нажатой, а указатель мыши находится в активной области. Допустимые значения опции: ключ /N — не выделяется; /I — инвертируется цвет содержания; /O — инвертируется цвет границы; /P — имитируется вдавливание (неплохо смотрится на цветном или сером фоне). По умолчанию действует ключ /I. Следующие ссылки демонстрируют действие ключа /I, ключа /O, ключа /P.

2.6. Закладки

Опция булевого типа

```
bookmarks
```

определяет, будут или не будут созданы закладки для документа. По умолчанию опция имеет значение true. Для того, **чтобы в закладках были видны русские буквы**, пакет **hyperref** надо подключать с опцией **unicode**. Вид закладок в Acrobat Reader регулируется следующими опциями:

```
bookmarksnumbered
```

задает нумерацию разделов; по умолчанию — false.

```
bookmarksopen
```

раскрывает полностью дерево закладок; по умолчанию — false.

В дополнение к докладам или отчетам теперь часто приходится делать электронные презентации, содержащие рисунки, таблицы и формулы. Современные электронные презентации должны отвечать следующим требованиям:

- Мобильность (возможность одинакового воспроизведения на разных компьютерах и платформах).
- Возможность использовать для создания свободно распространяемое программное обеспечение.
- Строгость, но «стильность» оформления.
- Наличие на слайдах рабочих ссылок на ресурсы Интернет и на локальные документы разных типов.
- Открытие сразу в полноэкранный режим.
- Наличие эффектов перехода и анимации объектов.
- Возможность использования звука и анимации на слайдах.
- Навигация по разделам и слайдам.
- Наличие титульного листа.

Высококачественную презентацию, удовлетворяющую первым двум требованиям, обеспечивает формат pdf. Файл презентации с расширением .pdf может быть получен разными средствами, например, путем экспорта презентации из офисного пакета. Но не все эффекты, работающие в офисном пакете, сохраняются при экспорте в pdf.

Средствами LaTeX возможна генерация файла презентации в формате pdf. При этом презентации, созданные при помощи современных пакетов LaTeX'а, успешно конкурируют с презентациями, созданным с помощью программы PowerPoint.

Для создания электронных презентаций в LaTeX'е существует ряд специальных пакетов, таких как Beamer, Prosper, PowerDot, PdfScreen, PPower4, PdfSlide и другие. При этом почти все приемы оформления текста в LaTeX будут работать и в презентациях, а значит из статьи или отчета pdf-файл презентации сделать очень просто.

Независимо от конкретного пакета, для получения pdf-файла презентации необходимо:

- Дважды обработать исходный файл имяфайла.tex при помощи pdflatex, т.е. выполнить дважды команду pdflatex имяфайла.

- Если в исходном файле присутствует postscript графика, то имяфайла.tex обрабатывается при помощи latex, полученный dvi-файл командой dvips конвертируется в имяфайла.ps, а уже затем в файл с расширением pdf (для этого используется команда ps2pdf). Цепочка latex-dvips-ps2pdf позволяет корректно вставить рисунки eps-формата и рисунки, выполненные с помощью пакета pstricks.

1. Пакет Beamer

Молодой (чуть более трех лет) пакет beamer разработан Тиллем Тантау (Till Tantau) и очень широко распространен в Америке, где де-факто стал стандартом для создания презентаций. В нем заложено огромное количество возможностей. В beamer возможно задание эффектов «проявления» текста на слайдах, эффектов анимации и перехода слайдов, а также большого числа вариантов оформления с прекрасным дизайном. Кроме того, в beamer при помощи пакета hyperref обеспечиваются ссылки как на объекты слайдов, так и на внешние объекты. Пакет beamer почти идеален для включения в презентацию анимации; музыка, кино и другие мультимедиа файлы легко присоединяются к создаваемому файлу.

Исходная документация по пакету Beamer занимает около 214 листов. После установки полной версии Miktex 2.7 ее можно найти в директориях Miktex2.7/doc/latex/beamer и Miktex2.7/doc/latex/powerdot.

- Пример типичной структуры презентации в `beamer`

1. Титульный лист.
2. Слайд с оглавлением по разделам.
3. Два раздела.
4. В одном разделе -- слайд со ссылками на слайды раздела, слайд с «проявлением» объектов и слайд с «выезжающей» таблицей, для которой имеются эффекты перехода;
5. В другом разделе — слайд со ссылками на слайды раздела, слайд со ссылками на различные документы (локальный `html`, рисунок, электронную таблицу, веб-страницу и внешний звуковой файл) и слайд с формулами, «проявляющимися» по щелчку мыши. Кроме того, для последнего слайда задан «эффект перехода» в виде «открывающихся жалюзей».

```
\documentclass[12pt]{beamer}
\usepackage[T2A]{fontenc}
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
\usepackage{graphicx}
%
\hyperbaseurl{.}
\hypersetup{pdfpagemode=FullScreen,colorlinks=true}
%
\setbeamercovered{transparent}
\usecolortheme{rose}
\usetheme{JuanLesPins}
\useinnertheme{rounded}
\useoutertheme{smoothbars}%
%
\title{Некоторые возможности \texttt{beamer}}
\date{октябрь 2008}
\author{\small О.В. Знаменская}
\institute{Сибирский федеральный университет}
\logo{\includegraphics[width=1cm]{logo}}
```

- Преамбула LaTeX документа.

Первые три строки определяют класс документа и возможности использования русского языка и графики.

Класс `beamer` загружает пакет `hyperref` по умолчанию. Но если в документе планируется использовать этот пакет с установками, отличными от стандартных, то их следует передать в качестве необязательного параметра команды выбора класса. Следующие две строки настраивают пакет `hyperref`. А именно, текущий каталог (т.е. каталог с презентацией) объявляется базовым, так что ссылки могут задаваться относительно него. Кроме того, объявляется открытие презентации в полноэкранном режиме и выделение ссылок разными цветами, в зависимости от типа ссылки.

Следующие пять строк определяют стиль оформления презентации.

Команда `\setbeamercovered{transparent}` обеспечивает «проявление» элементов презентации на слайде, следующие команды определяют цветовую гамму и стиль оформления слайдов. Стили презентации в данном пакете шифруются именем университета, в стиле и цветах которого предполагается оформить презентацию.

Последние строки в преамбуле задают содержание титульного листа(если не ставить дату, то `beamer` автоматически подставит текущую дату). Команда `\logo` печатает на слайдах нужный автору логотип (кафедры, факультета, университета или конференции).

- Выбор схемы презентации.

В пакете `beamer` можно выбрать *схему презентации, цветовую гамму, внешнюю и внутреннюю структуру*. Сочетание этих четырех параметров дает возможность получать различные визуальные эффекты.

Несколько возможных схем:

```
\usetheme{Warsaw}        \usetheme{Szeged}
\usetheme{Antibes}       \usetheme{JuanLesPins}
\usetheme{Montpellier}   \usetheme{Berkeley}
\usetheme{Boadilla}      \usetheme{PaloAlto}
\usetheme{CambridgeUS}
```

Несколько возможных цветовых гамм:

```
\usecolortheme[named=SeaGreen]{structure}
\usecolortheme{albatross}    \usecolortheme{crane}
\usecolortheme{wolverine}    \usecolortheme{sidebartab}
```

Внешняя структура:

```
\useoutnertheme{default}    \useoutnertheme{infolines}
\useoutertheme{shadow}      \useoutertheme{split}
\useoutertheme{smoothbars}  \useoutnertheme{sidebar}
```

Внутренняя структура:

```
\useinnertheme{default}     \useinnertheme{circles}
\useinnertheme{rectangles}  \useinnertheme[shadow]{rounded}
\useinnertheme{inmargin}
```

Можно начать с такой комбинации параметров:

```
\usecolortheme{rose}      \usetheme{Warsaw}
\useinnertheme{rounded}  \useoutertheme{shadow}
```

Внутренняя структура определяет оформление перечней, списков, блоков, теорем и таблиц содержания. В квадратных скобках могут быть перечислены дополнительные параметры. Команды `\useinnertheme` и `\useoutnertheme` можно не использовать, в этом случае структура будет формироваться по умолчанию.

В документации по `beamer` можно найти полный перечень вариантов оформления презентаций.

- Описание структуры презентации.

Сначала целесообразно задавать общую структуру документа (зарезервировать необходимое число слайдов и необходимое число разделов), а уже потом приступать к их наполнению содержанием и определению эффектов.

Структура презентации из двух разделов по два слайда в каждом, с отдельным слайдом содержания и титульной страницей, будет определяться в файле `имяфайла.tex` следующим образом:

```
\begin{document}
\begin{frame}\titlepage\end{frame}
\begin{frame}
\frametitle{Содержание}
\tableofcontents
\end{frame}
\section{Раздел1}
\begin{frame}
\frametitle{Содержание раздела1}%позже будет
\end{frame}
\begin{frame}[label=p1s1]
\frametitle{Слайд1p1}%будет содержание слайда
\end{frame}
```

```
\begin{frame}[label=p1s2]
\frametitle{Слайд2p1}%будет содержание слайда
\end{frame}
\section{Раздел2}
\begin{frame}
\frametitle{Содержание раздела2}%позже будет
\end{frame}
\begin{frame}[label=p2s1]
\frametitle{Слайд1p2}%будет содержание слайда
\end{frame}
\begin{frame}[label=p2s2]
\frametitle{Слайд2p2}%будет содержание слайда
\end{frame}
\end{document}
```

- Слайд со ссылками на слайды.

```
\section{Раздел 2}
\begin{frame}
\frametitle{Содержание раздела 2}
\begin{itemize}
\item \hyperlink{p2s1}{Слайд\,2.1: гиперссылки}
\item \hyperlink{p2s2}{Слайд\,2.2: формулы и
эффекты перехода}
\end{itemize}
\end{frame}
```

- Использование overlay и multimedia эффектов внутри текста.

```
\begin{frame}
Основной текст
\begin{itemize}
\only<1>{\item Первый элемент списка}
\only<2>{\item Второй элемент списка}
\only<3>{\item Третий элемент списка}
\end{itemize}
\end{frame}
```

```
\begin{enumerate}[<+->]
\item{ Первый элемент списка }
\item{ Второй элемент списка }
\item{ Третий элемент списка }
\item{ Четвертый элемент списка}
\end{enumerate}
```

- Слайд с формулами и эффектами перехода.

- Слайд с «проявлением» текста.

```
\begin{frame}[label=p1s1]
\frametitle{Проявление текста}
\pause
Этот текст <<проявляется>> после
первого щелчка мышью \\
\pause
Этот после второго
\end{frame}
```

```
\begin{frame}[label=p2s2]
\frametitle{Формулы и эффекты перехода}
\transblindshorizontal
%
\begin{block}{Логарифмическое отображение  Гаусса}
{\small \begin{equation*}
    \gamma: \, V \rightarrow \mathbb{CP}_{n-1} , \ \
    (z_1, \dots, z_n) \longmapsto (z_1 P^{\prime}{z_1}:
    \dots : z_n {P^{\prime}}{z_n}),
  \end{equation*}
    где $P^{\prime}{z_j} = \partial P / \partial z_j$.}
\end{block}
\pause
\begin{block}{Моментное отображение
$\mu: \mathbb{CP}_{n}\rightarrow \Sigma_n$}
{\small \begin{equation*}
    (Z_0:\dots:Z_n)\longrightarrow
    \frac{(|Z_0|,\dots,|Z_n|)}{|Z_0|+\dots+|Z_n|},
     \end{equation*}
 где $\Sigma_n=\bigg \{t\in\mathbb R^{n+1}:
 t_j\geq 0, \sum\limits_{j=0}^n t_j=1\bigg \}$
 --- стандартный симплекс.}
\end{block}
\end{frame}
```

- Слайд с «выползающей» таблицей.

```
\begin{frame}[label=p1s2]
\frametitle{<<Выползающая>> таблица}
\only<1>{
\begin{beamerboxesrounded}[shadow=true,width=7.5cm]
{Блок с таблицей}
\begin{tabular}{|r|c|l|} \hline
\textbf{Столбец1}&\textbf{Столбец2}&\textbf{Столбец3}\\
\hline Данные11 & Данные21 & Данные31 \\ \hline
\end{tabular}
\end{beamerboxesrounded}
}
\only<2>{
\begin{beamerboxesrounded}[shadow=true,width=7.5cm]
{Блок с таблицей}
\begin{tabular}{|r|c|l|} \hline
\textbf{Столбец1}&\textbf{Столбец2}&\textbf{Столбец3}\\
\hline Данные11 & Данные21 & Данные31 \\
\hline Данные12 & Данные22 & Данные32 \\ \hline
\end{tabular}
\end{beamerboxesrounded}
}
```

- Слайд с «выползающей» таблицей.

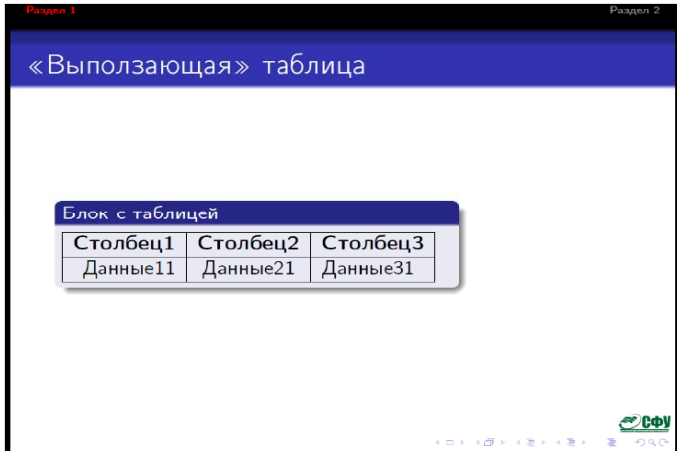
```
\only<3>{
\begin{beamerboxesrounded}[shadow=true,width=7.5cm]
{Блок с таблицей}
\begin{tabular}{|r|c|l|} \hline
\textbf{Столбец1}&\textbf{Столбец2}&\textbf{Столбец3}\\
\hline Данные11 & Данные21 & Данные31 \\
\hline Данные12 & Данные22 & Данные32 \\
\hline Данные13 & Данные23 & Данные33 \\
\hline
\end{tabular}
\end{beamerboxesrounded}
\beamerbutton{Кнопочка!
(Возвращает к содержанию Раздела 1)}
}\end{frame}
```

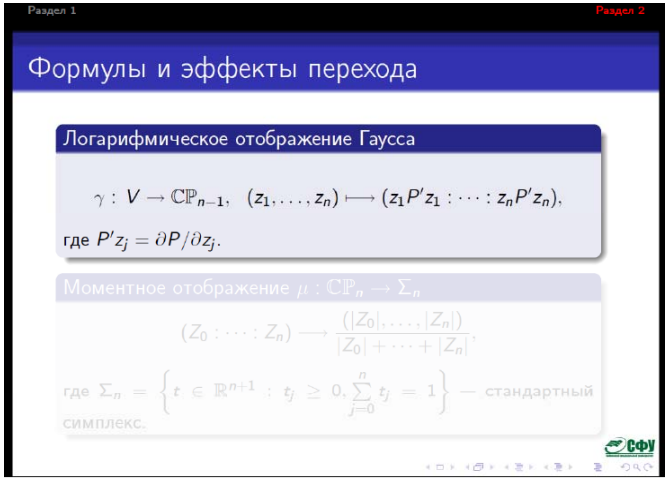
- Слайд с гиперссылками.

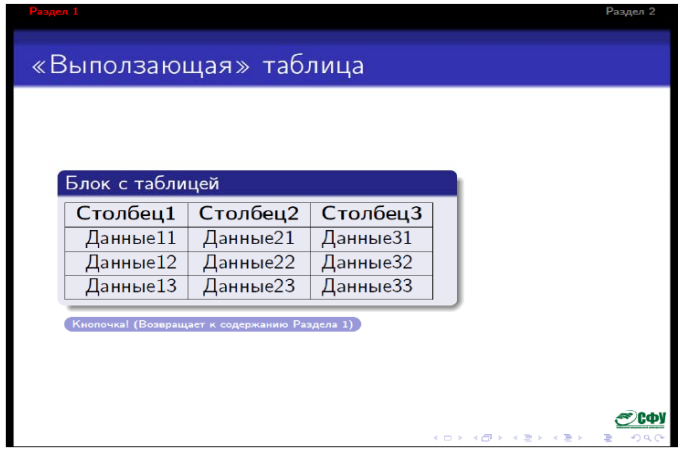
```
\begin{frame}[label=p2s1]
\frametitle{Гиперссылки}
\begin{enumerate}
\item \href{run:html/sample.html}%
{Локальный HTMLдокумент}
\item \href{run:images/sample.png}%
{Картинка}
\item \href{run:docs/sample.ods}%
{Электронная таблица в OpenOffice}
\item \href{http://ice.spb.ru}%
{Ссылка на сайт}
\item \href{run:sounds/sample.ogg}%
{Аудиоролик}
\end{enumerate}
\end{frame}
```

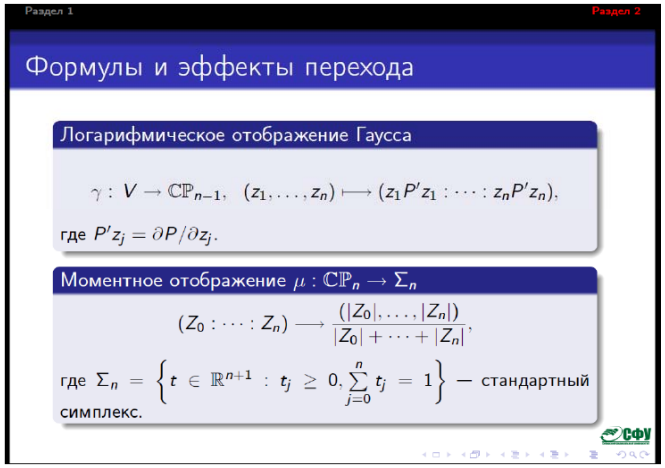
Модуль 3: Программирование презентационных эффектов

Лекция 21: Создание презентаций в LaTeX'е









2. Пакет PowerDot

```
\documentclass[size=12pt,style=fyma,mode=present,%
paper=a4paper,clock]{powerdot}
\usepackage[english]{babel}
\usepackage{amsfonts,amssymb,amsmath,mathrsfs}
\usepackage{listings}
\usepackage{pstricks,pst-node,pst-text,pst-3d,pst-grad}
\usepackage{moreverb,epsfig,color,subfigure}
\definecolor{brred}{rgb}{1.,.1,.1}
\title{PowerDot example}
\author{Author}
```

Необязательные аргументы команды `\documentclass`:

- `size=12pt` — размер шрифта (для презентации довольно маленький);
- `style=fyma` — определяет форму и цвета́ презентации. Вместо `fyma` можно использовать стили `default`, `simple`, `tycja`, `ikeda`, `ciment`, `elcolors`, `aggie`, `husky`, `sailor`, `upen`, `bframe horatio`, `paintings`, `klope`, `jefka`, `pazik`;
- `mode=present` — опция используется для вывода презентации на экран. Для печати и заметок используются опции `mode=print` и `mode=handout`;
- `paper=a4paper` — задает размер изображения, возможные варианты `paper=screen` и `paper=letterpaper`;
- clock — появление электронных часов в правом верхнем углу.

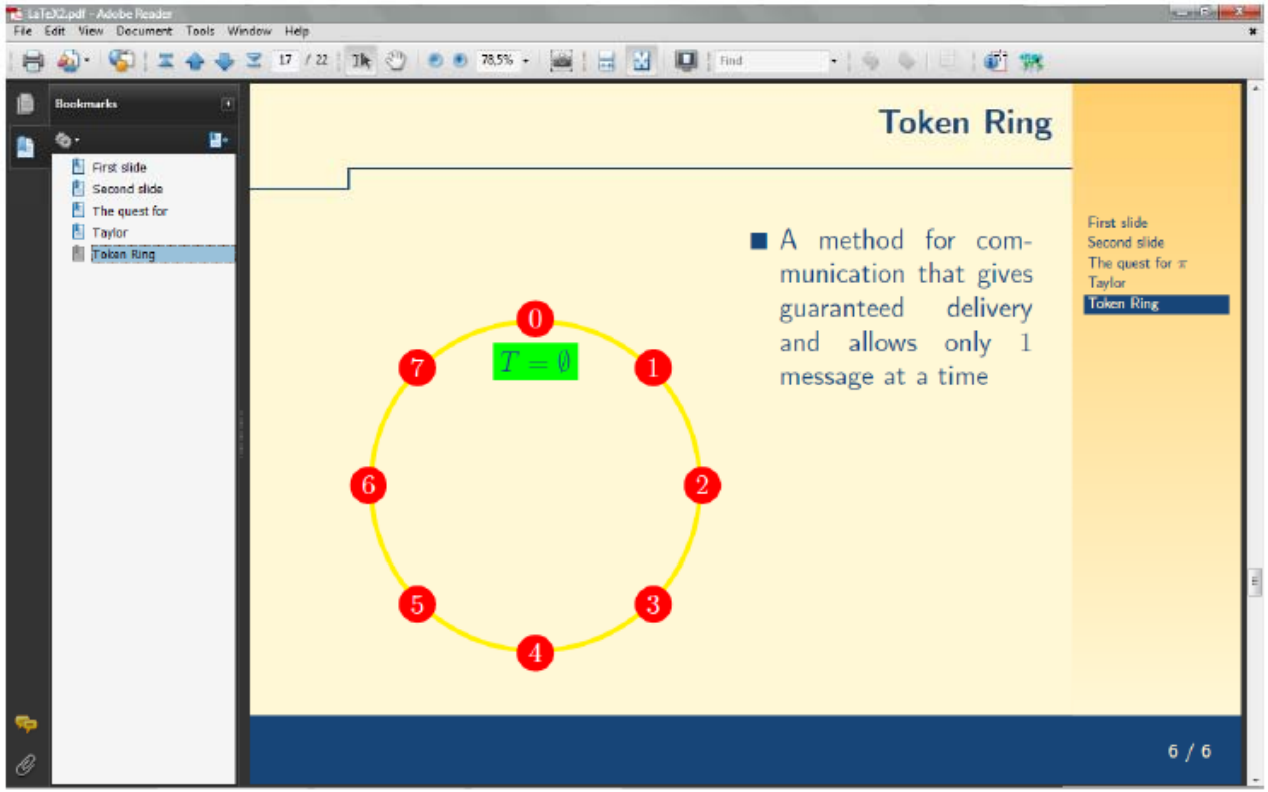
\begin{slide} ... \end{slide}

```
\begin{document}
\maketitle
\begin{slide}{First slide}
Hello \LaTeX!!!
\end{slide}
\begin{slide}[trans=Glitter/Di0]{Second slide}
It's my second slide.
\end{slide}
```

По умолчанию, все заголовки слайдов помещаются слева на экране для быстрого перехода между ними. Если необходимо использовать всю ширину экрана, то вывод списка слайдов можно отменить, используя заголовок типа `\begin{slide}[toc= ,bm=  ]{First slide}`

Вот немного более сложный слайд:

```
\begin{slide}[trans=Wipe]{The quest for $\pi$}
\begin{itemize}
\item The following formula computes $8$ correct
digits per iteration (Ramanujan):
\pause
\item
\begin{small}
\begin{equation*}
{\color{red} \frac{1}{\pi}}=\sum_{n=0}^\infty
\frac{(\frac{1}{4})_n(\frac{2}{4})_n
(\frac{3}{4})_n}{n!^3}
\bigl(2\sqrt{2}(1103+26390n)\bigr)
\frac{1}{(99^2)^{2n+1}}
\end{equation*}
\end{small}
\end{itemize}
\end{slide}%
```



Для эксперимента можно менять стили презентации в преамбуле и другие опции.

1. Создание электронных тестовых заданий средствами acrotex

Набор пакетов acrotex для изготовления в LaTeX электронных документов в формате PDF, в том числе разного рода учебных и тестовых заданий, был создан Д.П. Стори (D. P. Story) в университете Акрон (Akron) в 1999-2000 гг. и обновлен летом 2007 г. (вся документация и самый новый дистрибутив находятся в свободном доступе на персональной странице acrotex: http://www.math.uakron.edu/~dpstory/webeq.html.

1.1. Пакеты и их подключение

Для того, чтобы сгенерировать электронные тесты разного типа средствами макропакета acrotex, предпоследняя версия которого входит в современный дистрибутив MikTeX2.7, нужно в преамбуле документа подключить два обязательных пакета: пакет web и пакет exerquiz, написав команды:

```
\usepackage[<опции>]{web}
\usepackage[<опции>]{exerquiz}
```

Пакет Web переопределяет команды \maketitle и \tableofcontents в более дружественные: он раскрашивает заголовки параграфов и вставляет \bullets (•) на уровне \subsubsection, что выглядит на экране более привлекательно. В пакет также включены дополнительно определенные устройства для навигации — полочка и иконки с направлениями.

Имеются (необязательные) опции для подключения наиболее употребимых драйверов: dvipsone, dvips, dvipdfm и pdftex. В случае обработки исходного файла при помощи pdflatex, указывается опция pdftex.

Опция language переопределяет некоторые зависящие от языка элементы пакета на другие языки. Ее допустимые значения: dutch, french, german, italian, norsk, russian spanish, dansk, polish, finnish, czech, catalan, and brazil.

Опция переформатирования для распечатки на бумаге: forpaper. Используя forpaper, можно сделать бумажную копию упражнений (см. справочное руководство Webeqman.pdf).

Пакет **Web** задуман для демонстрации на экране учебных лекций или научно-технических материалов на конференциях. Для этих целей достаточно декларации стандартного класса **article**.

Пример типичной шапки документа:

```
\documentclass{article}
\usepackage{amsmath}\usepackage{literat}
\usepackage[cp866]{inputenc}
\usepackage[russian]{babel}
\usepackage[pdftex,russian]{web}
\usepackage[pdftex]{exerquiz}
\title{Название документа}
\author{Автор}
```

```
\subject{Назначание}
\keywords{Ключевые слова}
\university{Название университета}
\email{email}
\version{версия}
\copyrightyears{дата}
```

Пакет exerquiz предназначен для изготовления интерактивных файлов в формате PDF, содержащих разного рода задания, тесты и упражнения. Он может быть также использован для изготовления электронных форм. Далее рассмотрим подробнее примеры применения окружений из этого пакета.

Окружение exercise пакета exerquiz облегчает изготовление электронных упражнений. Рассмотрим несколько шаблонов окружений, определенных в exerquiz.

Наиболее интересна *-разновидность окружения exercise, Ее использование Позволяет включить в упражнение нескольких частичных вопросов. Следующие примеры упражнений иллюстрируют применение этой опции:

```
\begin{exercise}*\label{ex:parts}
Предположим, что частица движется вдоль оси $s$ и
местоположение частицы движущейся вдоль оси $s$
в любое время  $t$ вычисляется по формуле $s=t^2-5t+1$.
```

```
\begin{parts}
\item[h]\label{item:part} Вычислите скорость
        $v$ частицы при любом значении $t$.
\begin{solution}
Здесь приводится решение первого задания
\end{solution}
\item Вычислите ускорение $a$ частицы при любом
      значении $t$.
\begin{solution}
Здесь приводится решение второго задания
\end{solution}
\end{parts}
\end{exercise}
```

Ссылки могут быть сделаны к определённым частям упражнения; например,

```
\hyperref[item:part]{Exercise~\ref*{ex:parts}%
(\ref*{item:part})}.
```

Лекция 22: Электронные тесты и генерация вариантов заданий

Окружение `shortquiz` пакета `acrotex` позволяет конструировать тестовые задания с *выбором вариантов ответов*. Приведем пример такой организации заданий, что нажатие мышью на правильный вариант отсылает к пояснению ответа, а нажатие на неправильный вариант не вызывает никакого действия:

```
\begin{shortquiz}
Можно ли вскипятить воду снегом?
\begin{answers}[qz:water1]{2}
\Ans1 Можно & \Ans0 Нельзя
\end{answers}
\begin{solution}
Можно <...>
\end{solution}
\end{shortquiz}
```

Вопрос Можно ли вскипятить воду снегом?

(a) Можно (b) Нельзя

1. METAPOST и TEX

Желание Дональда Э. Кнута добиться качественной печатной продукции привело к созданию не только языка программирования TEX, но и и языка METAFONT. Изначально METAFONT предназначался Д. Кнутом для создания шрифтов и результатом его работы был растр с изображением шрифта. Позднее на основе METAFONT'а Т. Лефрумом (Tom Leathrum) был создан пакет `mfpic`, который используя графические возможности METAFONT'а позволил выполнять довольно качественные рисунки.

Позже аспирант Д.Э.Кнута Джон Хобби (John Hobby) модифицировал METAFONTтаким образом, что результатом работы программы стала картинка в формате `EPS` (`Encusulated PostScript`), добавлена поддержка цвета.

METAPOST проектировался как простая программа, которую можно относительно быстро настроить под свои нужды. Он вполне обозрим и компактен. Многие программы, работающие с векторной графикой, такие как `gnuplot` и `xfig`, умеют экспортировать в METAPOST. Шрифты можно внедрить посредством программ `latex` и `dvips` с результатом в виде `eps`-файла или скрипта `mptopdf` с результатом в виде `pdf`-файла. Эти картинки уже можно использовать независимо любой программой, которая поддерживает эти векторные форматы.

С одной стороны, графика METAPOST может импортироваться в документы, набираемые в TEX'е. С другой стороны, METAPOST'у можно поручить набор текстовых элементов TEX'а, например, текстовых меток или математических формул в графике. Таким способом графика в METAPOST может легко принять стиль документа (шрифт, размер шрифта и т. п.) и соответствовать качеству его набора (использовать кернинг, лигатуры и т. п.). Это делает METAPOST идеальным инструментом для приготовления высококачественной графики для документов в среде TEX.

Мы рассмотрим первую сторону взаимодействия TEX–METAPOST: импорту графики METAPOSTв TEX с использованием пакета `mfpic`.

Для использования METAPOST необходимо приготовить входной файл с METAPOST-кодом и затем вызвать сам METAPOST при помощи, как правило, команды в форме

```
mp <имя файла>.
```

Входные файлы для METAPOST обычно имеют имена, заканчивающиеся на `.mp`, и эта часть имени может опускаться при вызове METAPOST.

- Например, для входного файла `foo.mp`

```
mp foo
```

вызовет METAPOST и произведет выходные файлы с именами типа `foo.1` и `foo.2`.

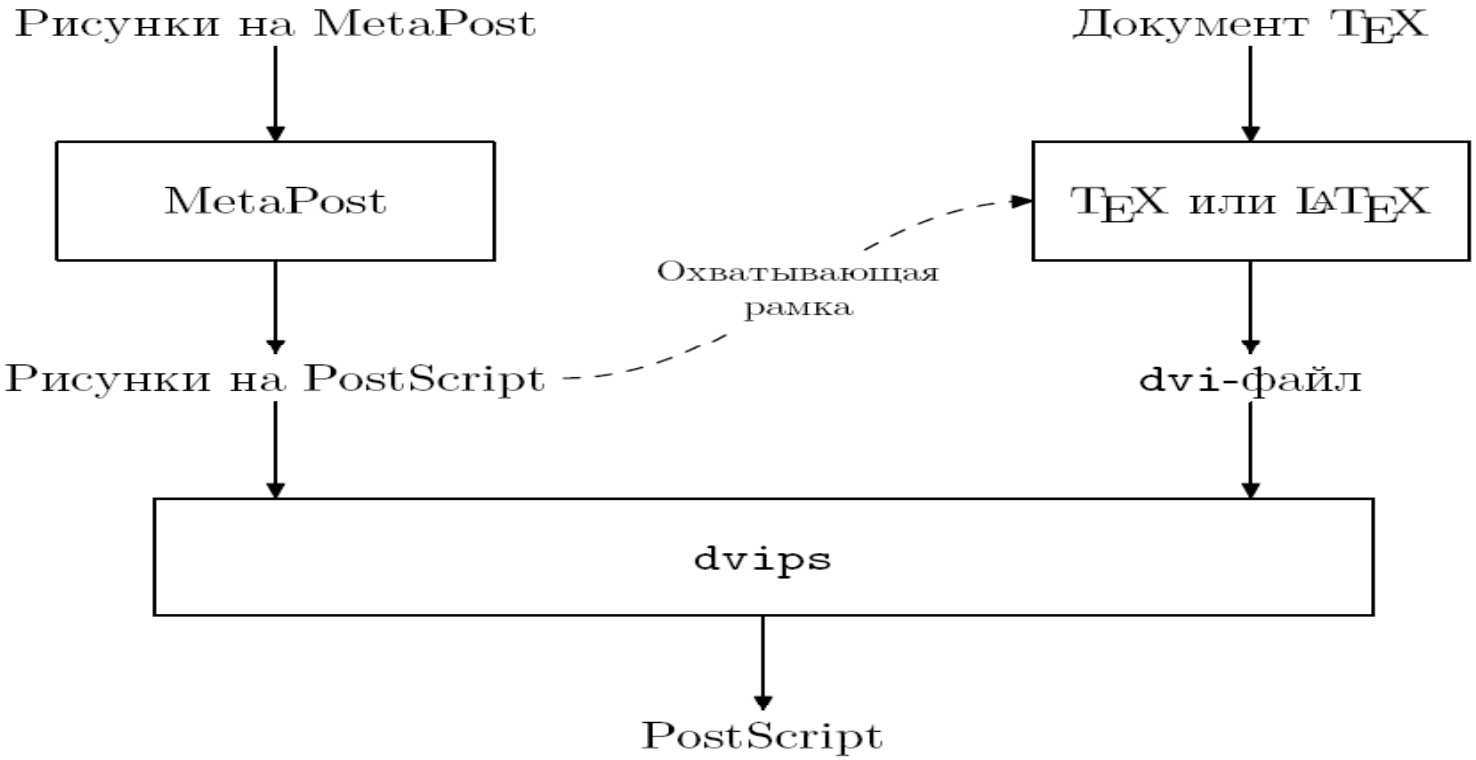


Рис. 23.1. Диаграмма обработки для документа с рисунками METAPOST

2. Кривые

В METAPOST'е основными объектами при рисовании являются кривые и прямые линии. Команда `draw` с разделенными `..` аргументами-точками рисует плавную кривую через эти точки. Например, посмотрите на результат

```
draw z0..z1..z2..z3..z4
```

после определения пяти точек таким образом:

```
z0 = (0,0);    z1 = (60,40);
z2 = (40,90); z3 = (10,70); z4 = (30,50);
```

Ниже показана кривая, проведенная через точки, помеченные от `z0` до `z4`.

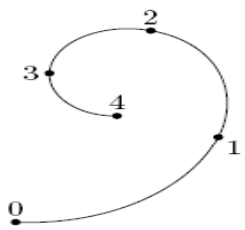


Рис. 23.2
Результат `draw z0..z1..z2..z3..z4`

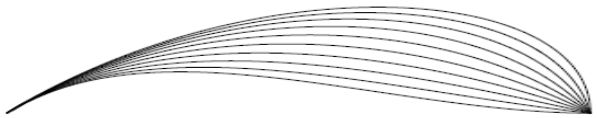


Рис. 23.6

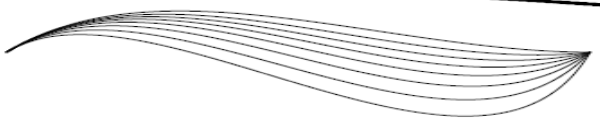


Рис. 23.7

Семейства кривых

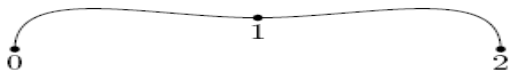
Инструкции MetaPost для генерации семейств кривых на рис. 23.6 и 23.7 соответственно:

```
beginfig(6)
for a=0 upto 9:
  draw (0,0){dir 45}..{dir -10a}(6cm,0);
endfor
endfig;
beginfig(7)
for a=0 upto 7:
  draw (0,0){dir 45}..{dir 10a}(6cm,0);
endfor
endfig;
```

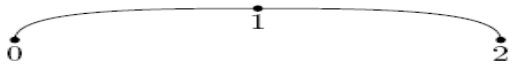
Вектора направлений, заданные в { }, могут быть любой длины и они могут как входить в точку, так и выходить из нее. Возможно даже в спецификации пути иметь оба направления для одной точки, до и после. Например, участок спецификации пути

```
..{dir 60}(10,0){up}..
```

произведет кривую с углом в (10, 0).



draw z0up..z1right..z2down



draw z0up..z1right..z2down

Рис. 23.8
Две команды draw и кривая-результат.

Лекция 24: Пакет MFPic

Первоначально mfpic был написан Т. Литрамом (Tom Leathrum) в 1992 году. Он состоял из пакета на языке TEX'а (который можно использовать вместе с LATEX'ом) и служебной библиотеки макросов на языке METAFONTá. В течение нескольких лет после этого были написаны и проверены различные версии. Многие разработки были выполнены другими авторами.

В 1997-99 годах пакет был дополнен библиотекой макросов на языке METAPOST'а (Dan Luecking). Это позволило, в частности, подключить цвет при рисовании. Мы приводим примеры работы с пакетом в этом последнем варианте.

При работе с пакетом MFPIC нужно начать с того, что в преамбуле входного файла необходимо загрузить дополнительный пакет (пакет mfpic с опцией metapost):

```
\begin{fmpage}{0.6\textwidth}
\usepackage[metafpost]{mfpic}
```

Далее дать команду на открытие файла для подготовки работы METAPOST'а и закрытие.

```
\opengraphsfile[<имя-файла>]
...  \closegraphsfile
```

1. Окружение mfpic

Окружение

```
\begin{mfpic}%
[<xscale>][<yscale>][<xneg>][<xpos>][<yneg>][<ypos>]
...
\ end{mfpic}
```

открывает и закрывает среду, в которой остальные макрокоманды, описанные ниже, будут иметь смысл.

Окружение `\mfpic` также устанавливает локальную систему координат для изображения. Параметры `<xscale>` и `<yscale>` устанавливают единичную длину в системе координат, как кратный размерности `\mfpicunit` (принятое по умолчанию значение которой равно 1 pt).

По крайней мере один параметр масштаба должен быть определен, но если определен только один параметр, то предполагается, что они равны.

Параметры `<xneg>` и `<xpos>` устанавливают нижний и верхний (соответственно) пределы для координатной оси x; аналогично, `<yneg>` и `<ypos>` устанавливают пределы для оси y.

1.1. Точки, линии, и прямоугольники

```
\pointdef{(name)}(x,y)
```

Эта макрокоманда присваивает символьное имя точке и ее координатам. То есть, если вы хотите часто использовать точку с координатами, например, (1.54553,45.4645), то намного удобнее будет писать просто a, заранее определив точку a таким образом:

```
\point{(<x0>,<y0>),(<x1>,<y1>),...}
```

Эта макрокоманда рисует круги с центрами в точках {(<x_0>,<y_0>)}, {(<x_1>,<y_1>)} и т.д. Диаметр окружностей определяется величиной pointsize. Значение pointsize по умолчанию 2 пункта.

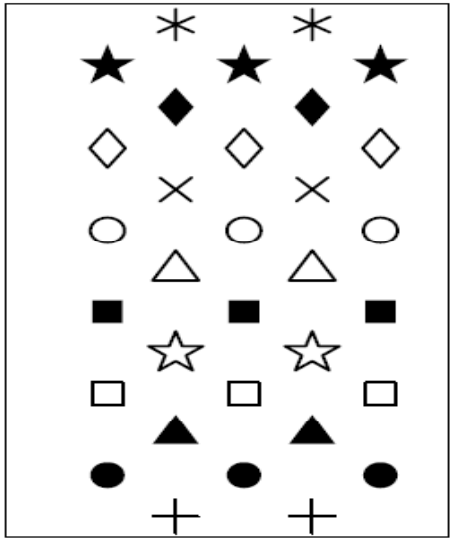
```
\plotsymbol[<size>]{<symbol>}{<p0>,<p1>,...}
```

Эта макрокоманде рисует маленькие фигурки в точках <p_0>, <p_1>, и т.д. Этим фигурам должно быть дано имена, такие как: Asterisk, Circle, Diamond, Square, Triangle, Star, SolidCircle, SolidDiamond, SolidSquare, SolidTriangle, SolidStar, Cross и Plus.

```
\begin{mfpic}[10][8]{0}{10}{0}{26}
\plotsymbol{Asterisk}{(3,25),(7,25)}
\plotsymbol{SolidStar}{(1,23),(5,23),(9,23)}
\plotsymbol{SolidDiamond}{(3,21),(7,21)}
\plotsymbol{Diamond}{(1,19),(5,19),(9,19)}
\plotsymbol{Cross}{(3,17),(7,17)}
\plotsymbol{Circle}{(1,15),(5,15),(9,15)}
\plotsymbol{Triangle}{(3,13),(7,13)}
\plotsymbol{SolidSquare}{(1,11),(5,11),(9,11)}
\plotsymbol{Star}{(3,9),(7,9)}
```

Лекция 24: Пакет MFPic

```
\plotsymbol{Square}{(1,7),(5,7),(9,7)}
\plotsymbol{SolidTriangle}{(3,5),(7,5)}
\plotsymbol{SolidCircle}{(1,3),(5,3),(9,3)}
\plotsymbol{Plus}{(3,1),(7,1)}
\end{mfpic}
```



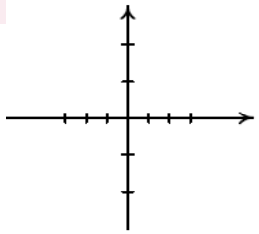
```
\polyline{(<x0>,<y0>),(<x1>,<y1>),...}
\lines{(<x0>,<y0>),(<x1>,<y1>),...}
```

Лекция 24: Пакет MFPic

```
\xmarks[<len>]{<numberlist>}
\tmarks[<len>]{<numberlist>}
\bmarks[<len>]{<numberlist>}
\ymarks[<len>]{<numberlist>}
\lmarks[<len>]{<numberlist>}
\rmarks[<len>]{<numberlist>}
\axismarks{<axis>}[<len>]{<numberlist>}
```

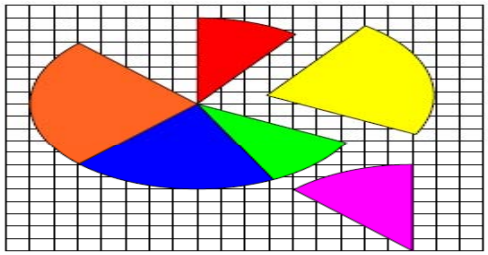
Эти макрокоманды ставят разделяющие метки на соответствующих осях в местах указанных в списке <numberlist>. Расстояние между метками определяется величиной <len>, и если этот необязательный параметр отсутствует, то расстояние определяется \hashlen. Величина \hashlen по умолчанию 4 пункта. Команда \axismarksx равносильна команде \xmarks, а также может использоваться для каждой из шести осей.

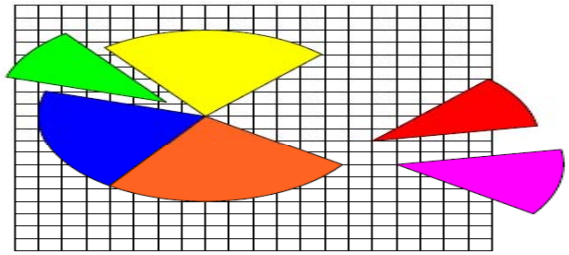
```
\begin{mfpic}[8]{-6}{6}{-6}{6}
\axes
\xmarks{-1,-2,-3,1,2,3}
\axismarks{y}{-2,2}
\end{mfpic}
```



2. Гистограмма и секторная диаграмма

```
\begin{mfpic}[7]{-10}{10}{-10}{10}
\piechart{(-2,2),7}{4,9,4,8,10,5}
\draw\gfill[(1,0,0)]\piewedge{1}
\draw\gfill[(1,1,0)]\piewedge[x3]{2}
\draw\gfill[(0,1,0)]\piewedge{3}
\draw\gfill[(0,0,1)]\piewedge{4}
\draw\gfill[Orange]\piewedge{5}
\draw\gfill[(1,0,1)]\piewedge[s(9,-12)]{6}
\end{mfpic}
```





```
\begin{mfpic}[7]{-10}{10}{-10}{10}
\piechart[a10]{(-2,1 ),7}{4,9,4,8,10,5}
\draw\gfill[(1,0,0)]\piewedge[s(7,-2)]{1}
\draw\gfill[(1,1,0)]\piewedge{2}
\draw\gfill[(0,1,0)]\piewedge[x2]{3}
\draw\gfill[(0,0,1)]\piewedge{4}
\draw\gfill[Orange]\piewedge{5}
\draw\gfill[(1,0,1)]\piewedge[m(6,-3)]{6}
\end{mfpic}
```

1. Типы данных

METAPOST имеет десять типов данных: числовой, для пар, для путей, трансформации, цвета (rgb-цвета), cmyk-цвета, строковый, логический, для картинок и тип пера.

Тип *pair* (пары) представляется как пара числовых количеств. Пары используются для задания координат в команде *draw*. Пары можно складывать, вычитать, использовать в выражениях усреднения, умножать и делить на числа.

Тип *path* (путей) — важные отдельные объекты, которые можно сохранять и изменять. Путь представляет прямую линию или кривую, определяемые параметрически.

Другой тип данных представляет произвольную *аффинную трансформацию* (transform). Трансформация может быть любой комбинацией вращений, масштабирований, наклонов и сдвигов.

Если `p = (px; py)` — это пара и `T` — это трансформация, то `p transformed T` — это пара вида

(tx + txxpx + txypy; ty + tyxpx + tyypy);

где 6 числовых количеств `(tx; ty; txx; txy; tyx; tyy)` определяют `T`.

Трансформации могут быть применимы к путям, рисункам, перьям и трансформациям.

Тип *цвета* (color) подобен типу пары, но он имеет три компоненты вместо двух и каждая компонента обычно находится в диапазоне от 0 до 1. Подобно парам, цвета могут складываться, вычитаться, использоваться в выражения усреднения, умножаться и делиться на числа. Цвета могут задаваться при помощи констант `black`, `white`, `red`, `green`, `blue` или явно заданными красной, зеленой и синей компонентами.

Черный — это `(0,0,0)` и белый — это `(1,1,1)`. Уровень серого, такой как `(.4,.4,.4)`, можно также задать как `0.4white`. Хотя цветовой переменной может быть любая упорядоченная тройка, при добавлении объекта к картинке METAPOST преобразует ее цвета обрезкой каждой цветовой компоненты до диапазона от 0 до 1. Например, METAPOST будет выводить цвет `(1,2,3)` как `(1,1,1)`.

METAPOST решает линейные уравнения с цветами таким же образом как и с парами. Тип `'rgbcolor'` (rgb-цвет) — это синоним типа `'color'` (цвет).

Тип *cmykcolor* (cmyk-цвета) подобен типу `color`, но имеет четыре компоненты вместо трех. Этот тип используется для задания цветов их зеленоголубой (cyan), пурпурнокрасной (magenta), желтой и черной компонентами. Из-за того, что cmyk-цвет использует краски вместо световых лучей, белый цвет будет выражаться как `(0,0,0,0)` и черный как `(0,0,0,1)`.

METAPOST имеет некоторые встроенные полезные numeric функции. Они включают: length, sqrt, sind, cosd, mlog, mexp, floor, angle, uniformdeviate, normaldeviate.

length x — абсолютное значение x;

sqrt x — квадратный корень x;

sind x — синус градуса x;

cosd x — косинус градуса x;

angle (x, y) — арктангенс y/x;

mlog x равный $256 \ln x$, где $\ln$ — логарифм по основанию e;

mexp x равный $e^{x/256}$.

floor x — самое большое целое число меньше или равное x.

METAPOST понимает ceiling, abs(то же самое, что и length), round, min и max.

Основные команды для рисования приведены в следующей таблице:

Имя	Объяснение
draw	Рисовать линию или картинку
drawarrow	Рисовать линию со стрелкой на конце
drawdblarrow	Рисовать линию со стрелками в обоих концах
fill	Заполнить циклический путь
filldraw	Рисовать циклический путь и заполнить его внутри
undraw	Стереть линию или рисунок
unfill	Стереть внутри замкнутого пути
unfilldraw	Стереть циклический путь и все внутри него
withcolor	Используй обычный цвет в команде рисования
withpen	Используй перо в команде рисования

6. Примеры

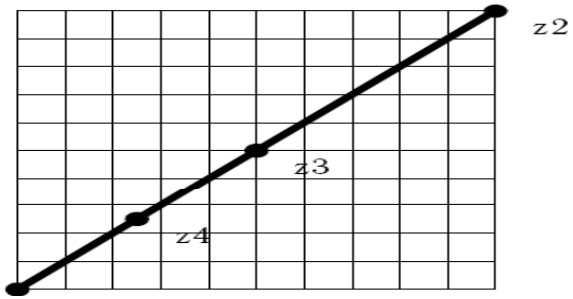
6.1. Определение точки

```
z1=(x1,y1);\ z2=(x2,y2);
```

Этой командой мы определяем точки $z1$ и $z2$, где $(x1, y1)$ и $(x2, y2)$ - координаты точек $z1$ и $z2$ соответственно. Заметьте, что после каждого командной строки необходимо ставить ; (точку с запятой). В дальнейшем уже не будет необходимости писать координаты этих точек.

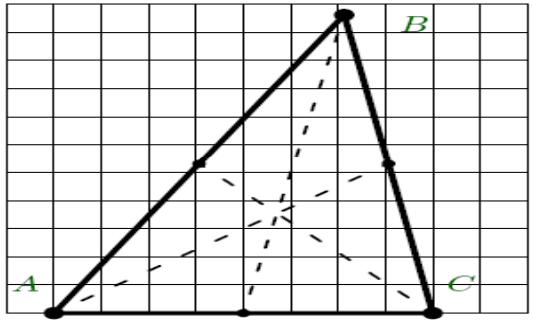
METAPOST легко выполняет задачу нахождения точки середины отрезка, 1/4 отрезка и так далее...

```
\begin{mfpic}[10]{0}{10}{0}{10}
\doaxes{lrtb}
\mfsrc{z1=(0,0); z2=(10,10);
z3=1/2[z1,z2]; z4=1/4[z1,z2];}
\point[2pt]{z1,z2}
\end{mfpic}
```



6.3. Построение медианы треугольника

Для того чтобы построить медианы необходимо найти середины сторон заданного треугольника и провести прямые к этим точкам из вершин треугольника.




```
\begin{mfpic}[12]{0}{11}{0}{11}
\mfsrc{z1=(1,0); z3=(9,0);
z2-z3=whatever*dir 100;
z2-z1=whatever*dir 60;}%
\lines{z1,z2,z3,z1} \point{z1,z2,z3}
\point{(0.5[z1,z2]),(0.5[z1,z3]),(0.5[z2,z3])}
\dashed\lines{z3,(0.5[z1,z2])}
\dashed\lines{z2,(0.5[z1,z3])}%
\dashed\lines{z1,(0.5[z2,z3])}%
\end{mfpic}%A=z1, B=z2, C=z3
```

1. Основные команды и приемы

Пути в METAPOST'е определяются как кусочные кубические кривые $(X(t), Y(t))$, где диапазон t от нуля до числа отрезков кривой. Другими словами, путь задается, как проходящий через последовательность точек, где $t = 0$ в первой точке, $t = 1$ в следующей, $t = 2$ в следующей и т. д.

Имеется несколько команд или операторов, которые позволяют извлекать информацию о кривых и применять ее для построений. Рассмотрим некоторые из них.

Оператор `intersectiontimes` — один из нескольких операторов, работающих с путями параметрически. Результатом

```
a intersectiontimes b
```

будет $(-1, -1)$, если пересечения нет; в противном случае, вы получите пару (t_a, t_b), где t_a — это время на пути `a`, когда он пересекает путь `b`, и t_b — это соответствующее время на пути `b`.

Например, предположим, что путь `a` обозначен тонкой линией на рис. 26.1 и путь `b` обозначен более толстой линией. Если метки показывают значения времени на путях, то пара значений времени, вычисленная в «`a intersectiontimes b`» должна быть одной из

$$(0.25, 1.77),\ (0.75, 1.40)\ (2.58, 0.24)$$

и какая из трех будет выбрана зависит от интерпретатора `MetaPost`. Точное правило выбора из многих точек пересечения достаточно сложное, но в данном примере вы получите $(0.25, 1.77)$. Меньшие значения времени предпочтительнее больших, так что (t_a, t_b) предпочтительнее, чем (t'_a, t'_b) пока $t_a < t'_a$ и $t_b < t'_b$. Когда нет простого способа минимизировать обе компоненты t_a и t_b, t_a получает больший приоритет, но правила становятся более сложными, когда между t_a и t'_a нет целых чисел. (Больше деталей см. в *The METAFONTbook*.)

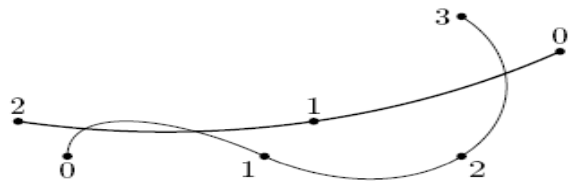


Рис. 26.1.
Два пересекающихся пути с отметками значений времени на каждом

Оператор `intersectiontimes` очень гибкий, потому что существует много всего, что можно сделать со значениями времени на пути.

Оператор `intersectionpoint` использует результат вычисления оператора `intersectiontimes`. Результатом

```
a intersectionpoint b
```

будет точка пересечения пути `a` пути `b`, соответствующая значениям времени, найденным оператором `intersectiontimes`.

2. Примеры

Пример 1

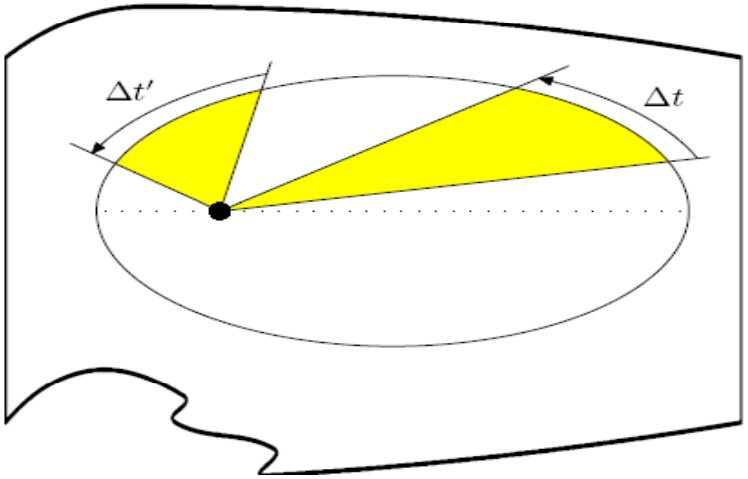
```
\mfpic[2]{0}{20}{0}{20}\mfsrc{u:=2cm;path p,q;
p:=(0,0){up}..(u,2u){dir 180};q:=(u,0){up}..(0,2u){up};
draw p withpen pencircle scaled 2bp;
draw subpath(0,.3) of q withpen pencircle scaled 1bp;
draw subpath(.45,1)of q withpen pencircle scaled 1bp;}
\endmfpic
```

Пример 2

```
\mfpic[1]{0}{100}{0}{100}%
\mfsrc{ u := 1cm; a := 6u; b := 3.5u;
 pair sun; sun := (-1.75u,0); path E[],
p[], area[];
 E1 = fullcircle xscaled a yscaled b;
  E2 = E1 scaled 1.1;
  p1 = sun -- (5u*dir(8) shifted sun);
   p2 = sun -- (4u*dir(28) shifted sun);
    p3 = sun -- (2u*dir(75) shifted sun);
     p4 = sun --(1.75u*dir(150) shifted sun);
      area1 = buildcycle(p1,E1,p2);
       area2 =buildcycle(p3,E1,p4);
 fill area1 withcolor red+green;
 fill area2 withcolor red+green;
  draw p1; draw p2; draw p3; draw p4; draw E1;
draw (-a/2,0)--(a/2,0) dashed withdots;
 draw sun withpen pencircle scaled 6bp;
 label.bot(btex Sun etex, sun);
numeric t[];
 t1 = ypart (p1 intersectiontimes E2);
 t2 = ypart (p2 intersectiontimes E2);
```

Лекция 26: Кривые в METAPOCT'e

```
t3 = ypart (p3 intersectiontimes E2);
t4 = ypart (p4 intersectiontimes E2);
drawarrow subpath (t1,t2) of E2;
drawarrow subpath (t3,t4) of E2;
label.urt(btex $\Delta t$ etex,
point((t1+t2)/2) of E2);
label.ulft(btex $\Delta t'$ etex,
point((t3+t4)/2) of E2);} \endmfpic
```



1. Пунктирные линии

Язык MetaPost предоставляет много способов изменять линии помимо простого изменения их толщины. Один такой способ в использовании пунктирных линий. Синтаксис для этого —

```
draw  p  dashed образец пунктира,
```

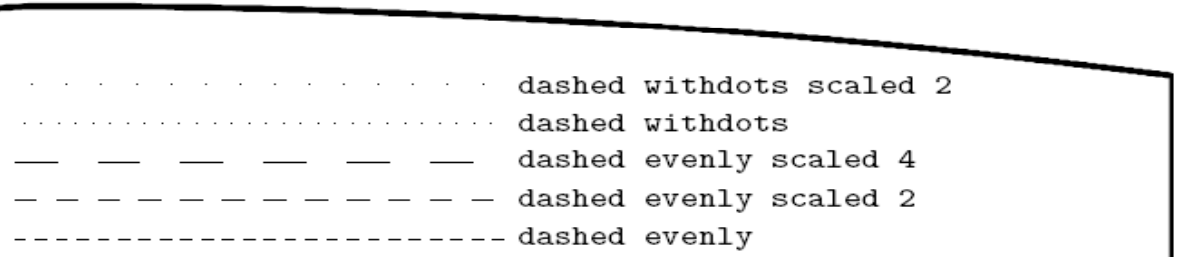


Рис. 27.1.

Пунктирные линии, помеченные образцом пунктира, используемым для их создания

2. Аффинные преобразования

Существует семь операторов трансформации с аргументом числом или парой :

$$
\begin{aligned}
(x,y)\ \texttt{shifted}\ (a,b) &= (x+a,\ y+b);\\
(x,y)\ \texttt{rotated}\ \theta &= (x\cos\theta - y\sin\theta,\ x\sin\theta + y\cos\theta);\\
(x,y)\ \texttt{slanted}\ a &= (x+ay,\ y);\\
(x,y)\ \texttt{scaled}\ a &= (ax,\ ay);\\
(x,y)\ \texttt{xscaled}\ a &= (ax,\ y);\\
(x,y)\ \texttt{yscaled}\ a &= (x,\ ay);\\
(x,y)\ \texttt{zscaled}\ (a,b) &= (ax-by,\ bx+ay).
\end{aligned}
$$

Большинство этих операторов объясняют сами себя, за исключением `zscaled`, о котором можно думать как о произведении комплексных чисел. Эффект `zscaled` (a,b) — это поворот и масштабирование так, что $(1,0)$ переходит в (a,b). Эффект от `rotated` θ во вращении на θ градусов против часовой стрелки.

Лекция 27: Стили рисования и модификация рисунков

```
path p[]; p1 = fullcircle scaled .6in;
z1=(.75in,0)=-z3; z2=directionpoint left of p1=-z4;
p2 = z1..z2..{curl1}z3..z4..{curl 1}cycle;
fill p2 withcolor .4[white,black]; unfill p1; draw p1;
transform T; z1 transformed T = z2;
z3 transformed T = z4;xxpart T=yypart T;
yxpart T=-xypart T; picture pic;
pic = currentpicture; for i=1 upto 2:
pic:=pic transformed T; draw pic;
endfor dotlabels.top(1,2,3); dotlabels.bot(4);
```

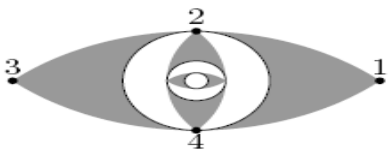


Рис. 27.3.
Код MetaPost и фрактальная картинка-результат

3. Стрелки

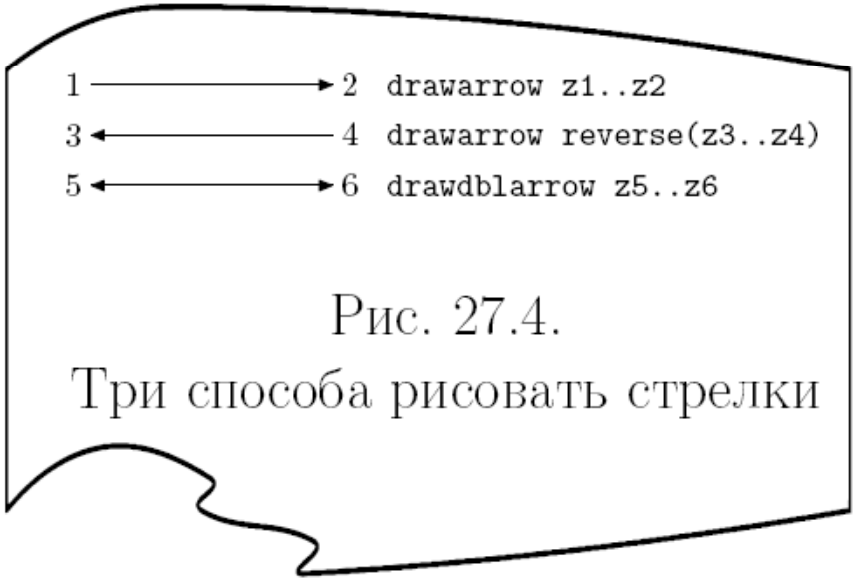


Рис. 27.4.
Три способа рисовать стрелки

Нарисованные стрелки, подобные тем, что на рис. 27.4, — это просто результат, когда вместо `draw` *выражение-путь* пишут

`drawarrow` *выражение-путь*.

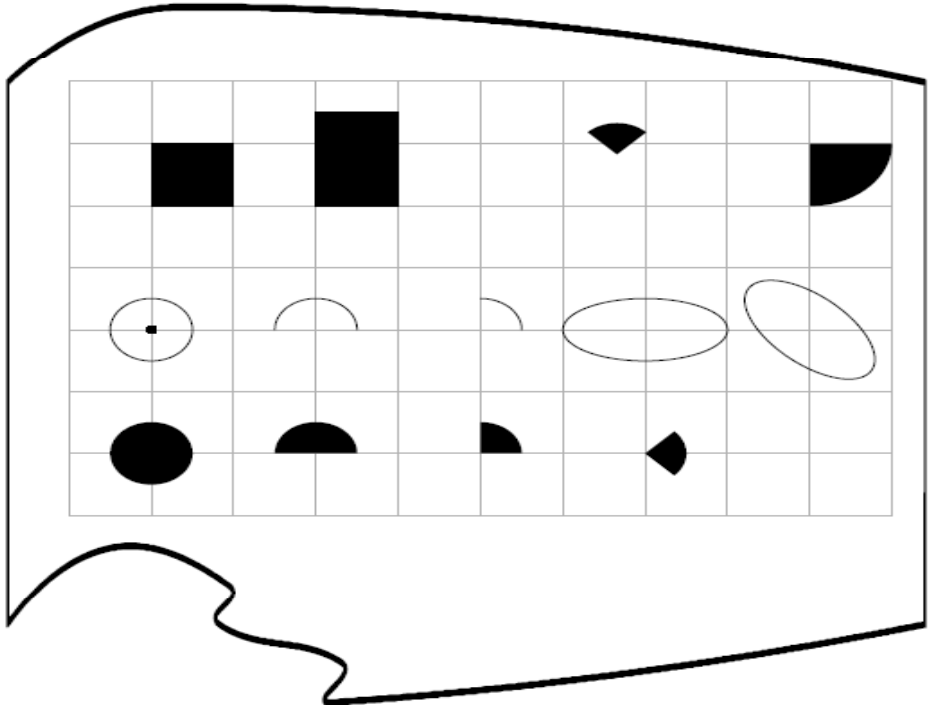
4. Стандартные кривые

Имеется некоторый стандартный набор кривых, которые можно использовать для рисования, применяя набор преобразований:

```
u := 24;
for i=-1 upto 9:
draw (i*u,4u)--(i*u,-3u) withcolor 0.7white;
endfor;
for i=-3 upto 4:
draw (-u,i*u)--(9u,i*u) withcolor 0.7white;
endfor;
dotlabel(,origin);
draw fullcircle scaled u;
draw halfcircle scaled u shifted (2u,0);
draw quartercircle scaled u shifted (4u,0);
draw fullcircle xscaled 2u yscaled u shifted (6u,0);
draw fullcircle xscaled 2u yscaled u
     rotated -45 shifted (8u,0);
fill fullcircle scaled u shifted (0,-2u);
fill halfcircle--cycle scaled u shifted (2u,-2u);
path quarterdisk; quarterdisk
     := quartercircle--origin--cycle;
fill quarterdisk scaled u shifted (4u,-2u);
```

Лекция 27: Стили рисования и модификация рисунков

```
fill quarterdisk scaled u rotated -45 shifted (6u,-2u);
fill quarterdisk scaled u shifted (6u,-2u) rotated 45;
fill quarterdisk rotated -90 scaled 2u shifted (8u,3u);
fill unitsquare scaled u shifted (0,2u);
fill unitsquare xscaled u yscaled 3/2u shifted (2u,2u);
```



1. Макросы

MetaPost имеет множество автоматически включаемых макросов, называемое макропакет Plain.

Макросы без аргументов очень просты. Определение макроса

```
def символический знак   = текст замены enddef
```

делает **символический знак** сокращением для **текста замены**, где **текст замены** может быть, в сущности, любой последовательностью знаков. Например:

```
def --- = ..tension infinity.. enddef.
```

Например, макрос **rotatedaround** определяется примерно так:

```
def rotatedaround(expr z, d) =
  shifted -z rotated d shifted z enddef;
```

и выражение

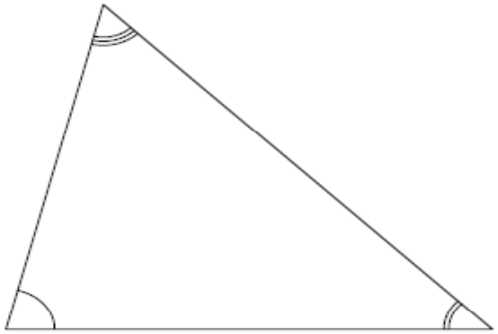
```
z1 rotatedaround (z2; 30)
```

просто раскрывается как

```
z1 shifted -z2 rotated 30 shifted z2.
```

- Еще пример макроса с параметрами:

```
 def draw_angle(expr A, O, B, n) =
    draw_angle_(A,O,B,5mm);
    if n>1 : draw_angle_(A,O,B,4.5mm); fi;
    if n>2 : draw_angle_(A,O,B,4mm);   fi;
    if n>3 : draw_angle_(A,O,B,5.5mm); fi;
  enddef;
  def draw_angle_(expr A,O,B,d) =
    draw (O + d*unitvector(A-O))
         { d*unitvector(A-O) rotated 90 }
         ..
         { d*unitvector(B-O) rotated 90 }
         (O + d*unitvector(B-O));
  enddef;
  pair A, B, C;
A := (0,0);
B := (3cm,0);
C := (1cm,2cm);
draw A--B--C--cycle;  draw_angle(B,A,C,1);
draw_angle(C,B,A,2);  draw_angle(A,C,B,3);
```



Слово `expr` в определении значит, что формальные параметры `z` и `d` могут быть произвольными выражениями. Этим параметрам следует соответствовать выражениям-парам, но MetaPost не проверяет этого сразу.

Из-за того, что MetaPost — это интерпретируемый язык, макросы с аргументами очень похожи на подпрограммы. Макросы MetaPost часто используются подобно подпрограммам, поэтому язык включает в себя программные концепции для такого использования. Эти концепции включают локальные переменные, циклы и условные команды.

1.1. Группировка

Группировка в MetaPost весьма важна для функций и локальных переменных.

Синтаксис для группы

```
begingroup {список команд}  endgroup
```

или

```
begingroup {список команд}{выражение}  endgroup
```

где `{список команд}` — это последовательность команд, за каждой из которых следует точка с запятой.

Имена переменных, начинающиеся с `x` или `y` являются локальными в том смысле, что они считаются неизвестными в начале каждой картинки и их значения забываются в конце каждой картинки. Следующий пример иллюстрирует как работает локальность.

```
x23 = 3.1;
beginfig(17);
    ⋮
y3a=1; x23=2;
    ⋮
endfig;
show x23, y3a;
```

Результат команды show

```
>> 3.1
>> y3a
```

показывает, что x23 возвращается к своему прежнему значению 3.1, а y3a совершенно неизвестно, как оно и было в beginfig(17).

Базовая идея макросов с параметрами в достижении большей гибкости передачей добавочной информации в макрос. Мы уже видели, что определения макросов могут иметь формальные параметры для выражений, задаваемых при вызове макроса. Например, определение

```
def rotatedaround(expr z, d)=текст замены  enddef
```

позволяет MetaPost-интерпретатору понимать вызовы макроса в форме

```
rotatedaround (выражение, выражение)
```

Ключевое слово `expr` в определении макроса значит, что параметры могут быть выражениями любого типа. Когда определение указывает `(expr z, d)`, то формальные параметры `z` и `d` ведут себя подобно переменным подходящего типа. Внутри *текста замены* они могут быть использованы в выражениях, как обычные переменные, но они не могут повторно декларироваться или получать новые значения. Нет ограничений на неизвестные или частично известные аргументы. Поэтому определение

```
def midpoint(expr a, b) = (.5[a,b]) enddef
```

прекрасно работает для неизвестных `a` и `b`. Уравнение, такое как

```
midpoint(z1,z2) = (1,1)
```

может быть использовано для определения `z1` и `z2`.

1. Интеграция текста и графики

MetaPost имеет несколько возможностей для включения меток и прочего текста в генерируемые им рисунки. Простейший способ сделать это в использовании команды `label`

```
label суффикс метки
      (выражение-строка или картинка выражение-пара);
```

Выражение-строка или картинка задает метку, а *выражение-пара* позицию для нее. *Суффикс метки* может быть **пустым**, что будет означать центрировать метку на заданных координатах. Если вы размечаете некоторые участки диаграммы, то вам вероятно понадобится слегка сместить метку, чтобы избежать накладки. Это иллюстрируется на рис. **??**, где метка `a` размещается над серединой указываемой линии, а метка `b` — слева от середины своей линии. Это достигается использованием `label.top` для метки `a` и `label.lft` для метки `b`, как показано на рисунке. **Суффикс метки** указывает позицию метки относительно заданных координат. Пол-

Рис.29.1 также иллюстрирует команду dotlabel.

```
beginfig(17);
a=.7in; b=.5in;
z0=(0,0);
z1=-z3=(a,0);
z2=-z4=(0,b);
draw z1..z2..z3..z4..cycle;
draw z1--z0--z2;
label.top( a , .5[z0,z1]);
label.lft( b , .5[z0,z2]);
dotlabel.bot( (0,0) , z0);
endfig;
```

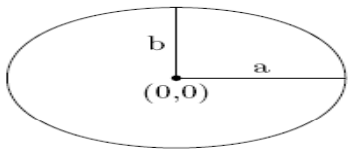
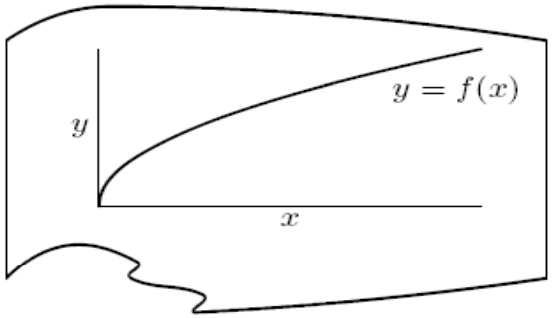


Рис. 29.1.
Код MetaPost и результат вывода

Еще пример размещения меток:




```
numeric u;
u = 1cm;
draw (0,2u)--(0,0)--(4u,0);
pickup pencircle scaled 1pt;
draw (0,0){up}
for i=1 upto 8: ..(i/2,sqrt(i/2))*u endfor;
label.lrt(btex $y=f(x)$ etex, (3,sqrt 3)*u);
label.bot(btex $x$ etex, (2u,0));
label.lft(btex $y$ etex, (0,u));
```

2. Взаимодействие METAPOST'а и LaTeX'а

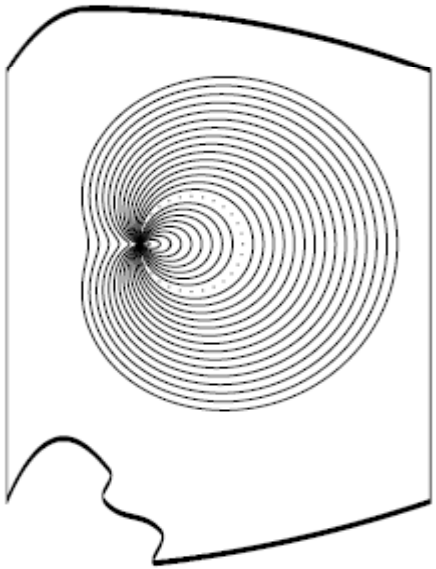
Возможны два способа использования графики в LaTeX'е. Первый — это использовать готовый файл с результатом обработки некоторого METAPOST-файла, который изготовлен вне системы LaTeX самостоятельно, либо взят из многочисленных источников разбросанных по различным сайтам в интернете. Тогда простейший файл для вставки графики, подготовленной METAPOST'ом и содержащейся в файле `foo.1`, в документ LaTeX'а может иметь вид:

```
\documentclass{article}
\usepackage{graphics}
\begin{document}
\includegraphics{foo.1}
\ end{document}
```

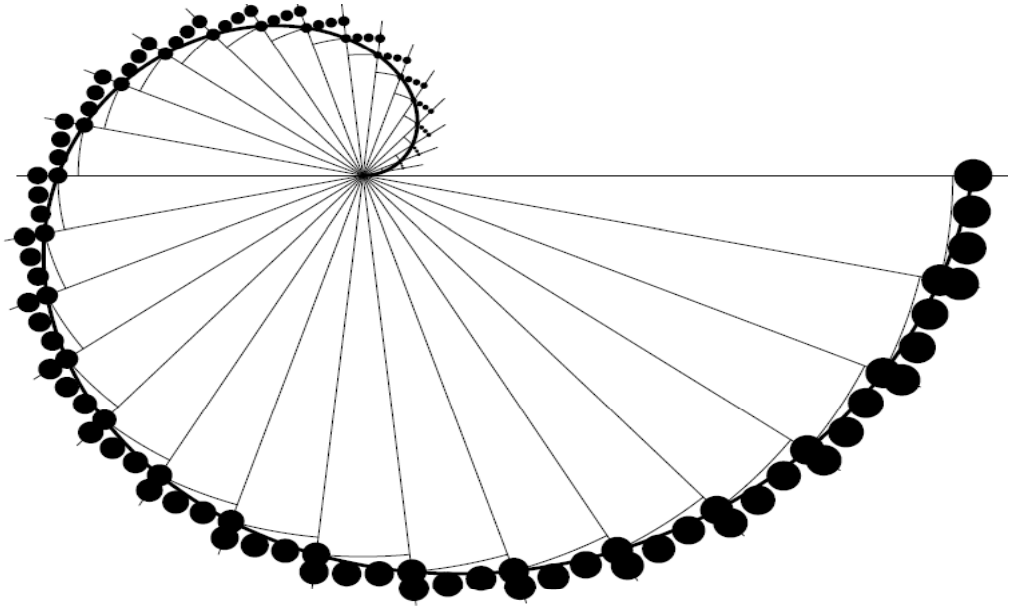
Лекция 29: Взаимодействие METAPOST'а и LaTeX'а

При подготовке графики в недрах самого LaTeX-файла с использованием METAPOST'а удобнее всего использовать пакет `mfpic` и команду `\mfsrc` для передачи данных в METAPOST. Наиболее простой способ — все графические построения проводить средствами пакета `mfpic`, а вычисления проводить METAPOST'ом передавая их через команду `\mfsrc`. Этим методом можно обойти сложность с выводом средствами METAPOST'а русскоязычного текста и сложных LaTeX-выражений (например, математических конструкций) через метки. Посмотрите на двух приведенных ниже примерах этот прием.

```
\begin{mfpic}[50]{-3}{3}{-3}{4}
\mfsrc{for n=1 upto 15:}
 \plrfcn{-180,180,10}{cosd(t)+n/10}
\mfsrc{endfor;} \pen{1pt}
 \dotted\plrfcn{-90,90,10}{cosd(t)}
\end{mfpic}
```



Лекция 29: Взаимодействие METAPOST'а и LaTeX'а



```
\begin{mfpic}[18]{-3.3}{6.5}{-5}{2.3}
\lines{(0,0),(cosd(12*1)*(0.2*1+0.4),
sind(12*1)*(0.2*1+0.4))}
\mfsrc{for n=2 upto 30:}\lines{(0,0),
(cosd(12*n)*(0.2*n+0.4),sind(12*n)*(0.2*n+0.4))}
\arc[p]{(0,0),12*(n-1),12*n,0.2*n-0.2}
\dotted[0.5*n,0.5*n]\arc[p]{(0,0),12*(n-1),12*n,0.2*n}
\mfsrc{endfor;}\pen{1.5pt}\plrfcn[s]{0,360,12}{t/60}
\ end{mfpic}
```

1. Поддержка трехмерной графики в LaTeX

В данной лекции, прежде всего, рассмотрим некоторые возможности пакета трехмерной иллюстративной графики mfpic3d, разработанного два года назад С.В.Знаменским на базе mfpic и perl, и уже эффективно применявшегося для подготовки к изданию книг по аналитической и дифференциальной геометрии. В настоящее время пакет активно совершенствуется. Эти картинки изготовлены с помощью mfpic3d:

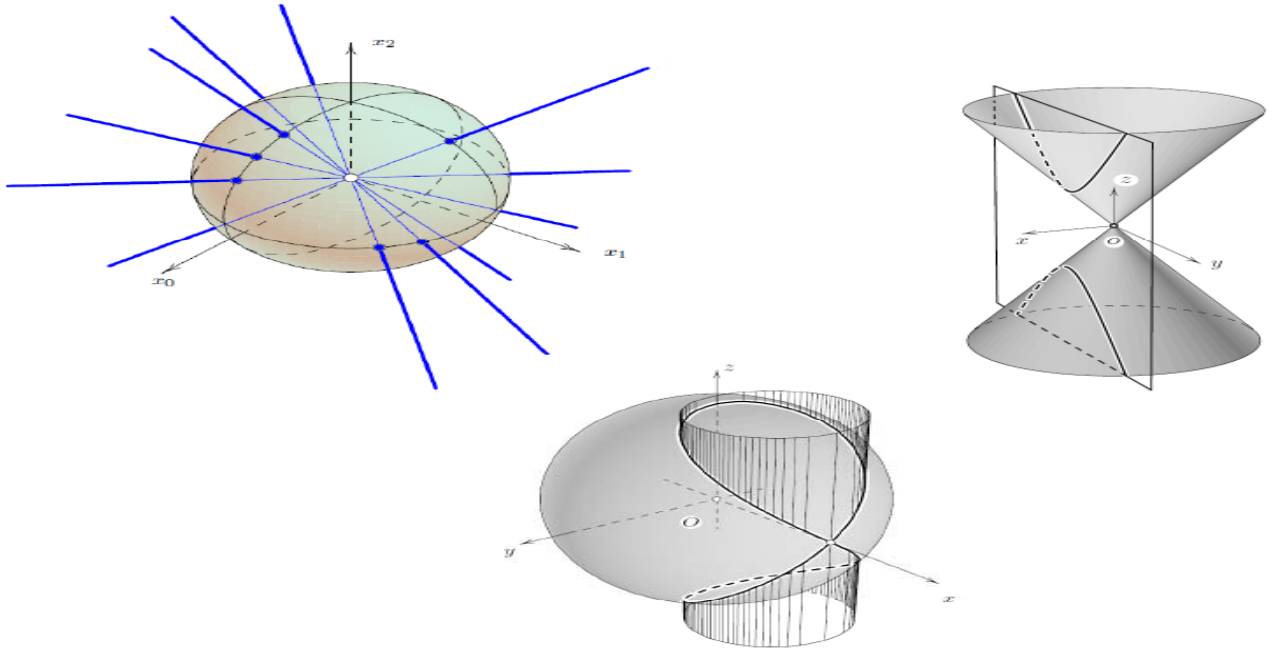


Рис 30.1. Примеры изображений в mfpic3d

1.1. Установка mfpic3d

Комплектация

Для того, чтобы воспользоваться возможностями пакета, на компьютере должен быть установлен MiKTEX версии не ниже шестой. В редактируемом файле `имяфайла.tex` с расширением `.tex` должен быть подключен пакет

```
\usepackage[metapost,truebbox,mplabels,dvips]{mfpic}
```

В преамбуле `имяфайла.tex` необходимо подключить стилевой файл `mfpic3d.sty`. Поскольку `mfpic3d` использует интерпретатор PERL, на компьютере должен быть установлен PERL, а файл `mfpic3d.pl` необходимо поместить в директорию, в которой TEX сможет его найти (например, в текущую рабочую директорию).

Для компиляции Файла необходим FAR или любой другой редактор, позволяющий работать с командной строкой.

Последовательность обработки

Обрабатывается, как обычно, основной файл `имяфайла.tex`, для этого в командной строке надо дать команду `latex имяфайла.tex`. После этого в рабочей директории появятся файлы: `mp1.mp , mp2.mp , ...` Их столько, сколько картинок содержится в файле `имяфайла.tex`.

Далее необходимо обработать эти файлы так, чтобы они стали «читабельны» для команды `mp.exe`. Для этого

- в командной строке дается команда: путь файла `perl.exe mfpic3d.pl mp1.mp` (стандартное расположение: `C:\PERL\bin\perl.exe mfpic3d.pl mp1.mp`), после чего в директории появится новый файл `mp1.mp.spl`;
- в командной строке дается команда: `makempx.exe mp1.mp` `mp1.mpx` (это стандартная процедура для обработки надписей `mplabels`), после чего в директории появится новый файл `mp1.mpx`.

Точка зрения

Далее необходимо задать точку зрения, т. е. угол под которым будет видна трехмерная картинка. Для этого служит команда

```
\ViewPoint \{a,b,c\}.
```

Например, если задать параметры a, b, c, равные, соответственно, 5, 0 и 0, то наблюдатель будет помещен на ось Ox в точку с координатами (5,0,0). Пример изменения точки зрения приведен на рис. 30.2.

Приведем некоторые картинки, изготовленные с помощью `mfpic3d`.

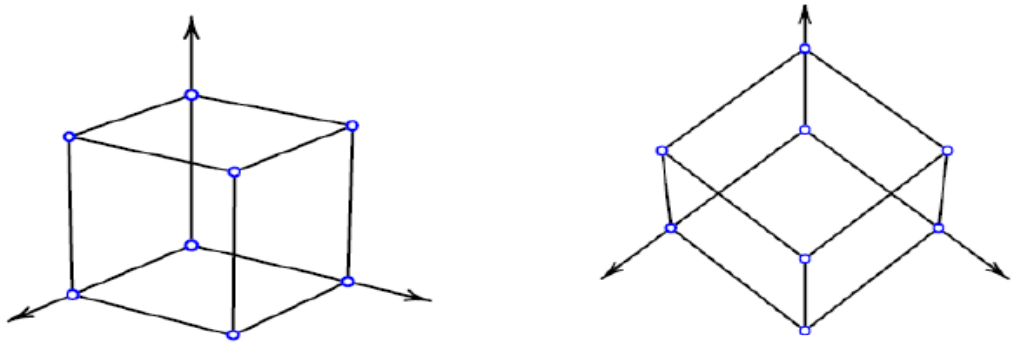


Рис 30.2 Куб: две точки зрения

Для первого куба точка зрения была задана командой `\ViewPoint {30,40,20}`, а для второго, соответственно, командой `\ViewPoint {20,20,50}`. Конечно, такой куб возможно нарисовать и средствами METAPOST, но пакет **mfpic3d** дает возможность изменять угол зрения на объект, меняя лишь аргументы команды `\ViewPoint`, тогда как в METAPOSTдля этого придется рисовать новый куб целиком. Остальные команды для рисования обоих кубов:

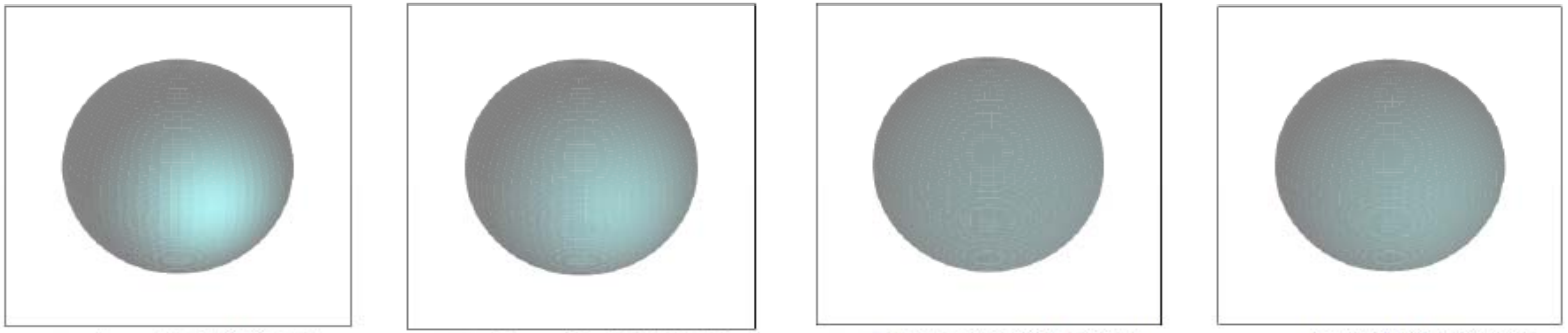
```
\mfsrc{z1=\(0,0,0\);z2=\(4,0,0\);z3=\(4,4,0\);
z4=\(0,4,0\);z5=\(0,0,4\);z6=\(4,0,4\);z7=\(0,4,4\);
z8=\(4,4,4\);z9=\(6,0,0\);z10=\(0,6,0\);z11=\(0,0,6\);}
\lines{z1,z2,z3,z4,z1,z5,z6,z8,z7,z5}\lines{z4,z7}
\lines{z8,z3}\lines{z6,z2}\arrow\lines{z1,z9}
\arrow\lines{z1,z10}\arrow\lines{z1,z11}
```

Подсветка поверхности

Двумерная поверхность (сфера, цилиндр, конус) подсвечивается, создается светотень, для этого нужно использовать команду

```
\Lamps{[x, y, z], [d, e, f]}
```

Параметры `x, y, z` задают пространственные координаты точки, где помещена лампа, а `d,e` и `f` отвечают за яркость цвета: `d` — красного, `e` — зеленого, `f` — синего.



Lamps [15,-15,0],[.2,.5,.5] Lamps [20,-20,0],[.2,.5,.5] Lamps [200,-200,0],[.2,.5,.5] Lamps [40,-40,0],[.2,.5,.5]

Рис 30.4. Подсветка поверхности сферы

Цвет поверхности

Для того чтобы выбрать цвет поверхности, используют команду

```
\SurfaceColors {[a, b, c],[a, b, c]},
```

в которой первая тройка чисел задает раскраску передней, лицевой части поверхности (лица), а вторая — внутренней, изнаночной части поверхности (изнанки). Здесь параметры `a`, `b` и `c` определяют, соответственно, интенсивность зеленого, красного и желтого цветов.

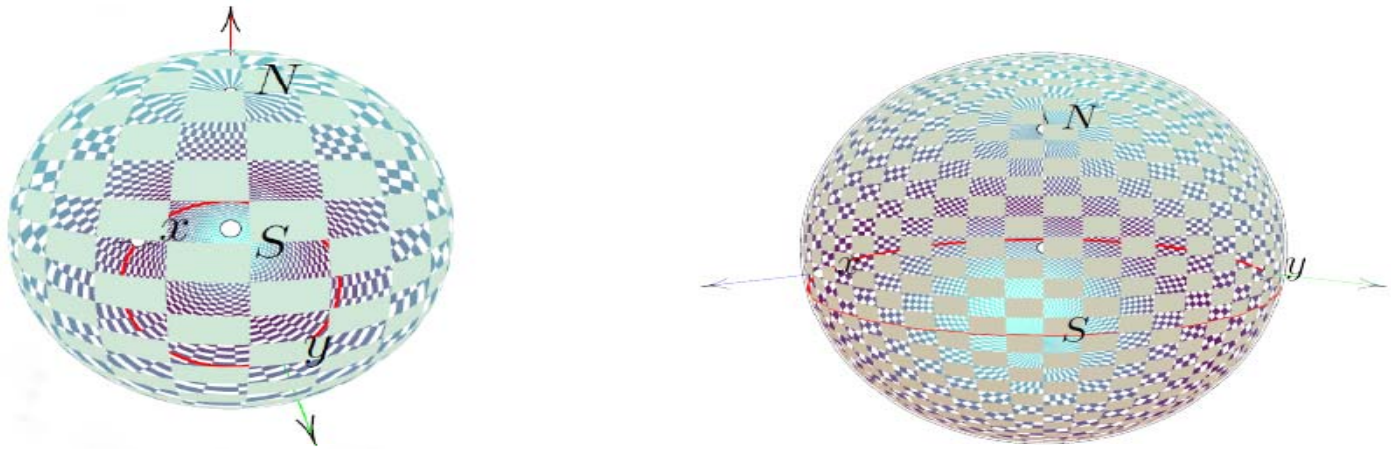


Рис 30.5. Подсветка и раскраска внутренней поверхности сферы