Государственное образовательное учреждение ВПО
Уральский государственный экономический университет

Ю. Б. Мельников

Справочник по командам LaTeX

e-mail: melnikov@k66.ru,
melnikov@r66.ru
сайты:
http://melnikov.k66.ru,
http://melnikov.web.ur.ru

Екатеринбург
2012

Ю.Б.Мельников

Некоторые команды LaTeX

A, B, C, D, E, F, G, H, I, J, K, L, M,
N, O, P, Q, R, S, T, U, V, W, X, Y, Z

I. Некоторые символы и буквы

I.1. Дефис, тире, §, №, Ё, ё, ударение, «...», “...”

Чтобы получить дефис «-», используется одиночный знак `-`.

Чтобы получить знак «–», используется «удвоенный минус» `--`.

Чтобы получить тире «—», используется «утроенный минус» `---`.

Знак параграфа § печатается командой `\S`.

Знак номера №: команда `\No` (лучше в математической моде). Учтите, что пробел после `\S` и `\No` подавляется: результатом компиляции `\S 5` и `\No 5` будет §5 и №5 (без пробела между 5 и `\S`, `\No`).

Буквы Ё и ё набираются как `\"{E}` и соответственно, `\"{e}`, поскольку команда `\"{...}` ставит двойную точку над буквой, обозначенной здесь троеточием.

Команда команда `\'{...}` ставит штрих над буквой, обозначенной здесь троеточием: `<<p\'{ы}ба>>` после компиляции даёт «ры́ба».

Кавычки: «ёлочкой» — `<<\"{e}лочкой>>`, “западные” — ``‘‘западные’’``.

I.2. Греческий алфавит

α	\alpha	β	\beta	γ	\gamma
δ	\delta	ϵ	\epsilon	ε	\varepsilon
ζ	\zeta	η	\eta	θ	\theta
ϑ	\vartheta	ι	\iota	κ	\kappa
λ	\lambda	μ	\mu	ν	\nu
ξ	\xi	π	\pi	ϖ	\varpi
ρ	\rho	ϱ	\varrho	σ	\sigma
ς	\varsigma	τ	\tau	υ	\upsilon
ϕ	\phi	φ	\varphi	χ	\chi
ψ	\psi	ω	\omega		

Γ	\Gamma	Δ	\Delta	Θ	\Theta
Λ	\Lambda	Ξ	\Xi	Π	\Pi
Σ	\Sigma	Υ	\Upsilon	Φ	\Phi
Ψ	\Psi	Ω	\Omega		

I.3. Готический алфавит

Буквы готического шрифта набираются в математической моде.

Переключение на готический шрифт происходит командой `\mathfrak`.

Например, в результате компиляции кода

`$\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$` получаем

$\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

А в результате компиляции кода

`$\mathfrak{abcdefghijklmnopqrstuvwxyz}$` получаем

$\mathfrak{abcdefghijklmnopqrstuvwxyz}$

I.3. Готический алфавит

Буквы готического шрифта набираются в математической моде.

Переключение на готический шрифт происходит командой `\mathfrak`.

Соответствие букв латинского и готического алфавитов:

$$\begin{array}{cccccccccccccccccccccccccc}
A & B & C & D & E & F & G & H & I & J & K & L & M & N & O & P & Q & R & S & T & U & V & W & X & Y & Z \\
\mathfrak{A} & \mathfrak{B} & \mathfrak{C} & \mathfrak{D} & \mathfrak{E} & \mathfrak{F} & \mathfrak{G} & \mathfrak{H} & \mathfrak{I} & \mathfrak{J} & \mathfrak{K} & \mathfrak{L} & \mathfrak{M} & \mathfrak{N} & \mathfrak{O} & \mathfrak{P} & \mathfrak{Q} & \mathfrak{R} & \mathfrak{S} & \mathfrak{T} & \mathfrak{U} & \mathfrak{V} & \mathfrak{W} & \mathfrak{X} & \mathfrak{Y} & \mathfrak{Z}
\end{array}$$

$$\begin{array}{cccccccccccccccccccccccccc}
a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\
\mathfrak{a} & \mathfrak{b} & \mathfrak{c} & \mathfrak{d} & \mathfrak{e} & \mathfrak{f} & \mathfrak{g} & \mathfrak{h} & \mathfrak{i} & \mathfrak{j} & \mathfrak{k} & \mathfrak{l} & \mathfrak{m} & \mathfrak{n} & \mathfrak{o} & \mathfrak{p} & \mathfrak{q} & \mathfrak{r} & \mathfrak{s} & \mathfrak{t} & \mathfrak{u} & \mathfrak{v} & \mathfrak{w} & \mathfrak{x} & \mathfrak{y} & \mathfrak{z}
\end{array}$$

I.4. Верхние и нижние индексы

Верхний и нижний индекс ставится ***только в математической моде.***

Верхний индекс ставится с помощью символа `^`. Например, код `$2^{x}$` в результате компиляции даёт 2^x.

Нижний индекс ставится с помощью символа `_`. Например, код `$2_{x}$` в результате компиляции даёт 2_x.

I.5. Некоторые элементарные функции

$\log_x y$	`\log_{x}{y}`	$\lg x$	`\lg{x}`	$\ln x$	`\ln{x}`
$\sin$	`\sin`	$\cos$	`\cos`	$\arcsin$	`\arcsin`
tg	`\tg`	ctg	`\ctg`	$\arccos$	`\arccos`
$\sqrt{}$	`\sqrt{...}`	arctg	`\arctg`	arcctg	`\arcctg`

I.6. Бинарные операции

$+$	+	$-$	-	$*$	*
$\pm$	\pm	$\mp$	\mp	$\times$	times
$\div$	\div	$\setminus$	\setminus	$\cdot$	\cdot
$\circ$	\circ	$\bullet$	\bullet	$\cap$	\cap
$\cup$	\cup	$\uplus$	\uplus	$\sqcap$	\sqcap
$\sqcup$	\sqcup	$\vee$	\vee	$\wedge$	\wedge
$\oplus$	\oplus	$\ominus$	\ominus	$\otimes$	\otimes
$\odot$	\odot	$\oslash$	\oslash	$\triangleleft$	\triangleleft
$\triangleright$	\triangleright	$\amalg$	\amalg	$\diamond$	\diamond
$\wr$	\wr	$\star$	\star	$\dagger$	\dagger
$\ddagger$	\ddagger	$\bigcirc$	\bigcirc	$\bigtriangleup$	\bigtriangleup
$\bigtriangledown$	\bigtriangledown				

I.7. Отношения

$<$	<	$>$	>	$=$	=	$\leqslant$	\leqslant
$:$	:	$\sim$	\sim	$\simeq$	\simeq	$\geqslant$	\geqslant
$\approx$	\approx	$\cong$	\cong	$\equiv$	\equiv		
$\ll$	\ll	$\gg$	\gg	$\doteq$	\doteq		
$\leqslant$	\le, \leq	$\geqslant$	\ge, \geq	$\in$	\in		
$\notin$	\notin	$\ni$	\ni	$\subset$	\subset		
$\subseteq$	\subseteq	$\supset$	\supset	$\supseteq$	\supseteq		
$\succ$	\succ	$\prec$	\prec	$\succeq$	\succeq		
$\preceq$	\preceq	$\asymp$	\asymp	$\sqsubseteq$	\sqsubseteq		
$\sqsupseteq$	\sqsupseteq	$\models$	\models	$\vdash$	\vdash		
$\perp$	\dashv	$\smile$	\smile	$\frown$	\frown		
$\mid$	\mid	$\bowtie$	\bowtie	$\bowtie$	\Join		
$\propto$	\propto	$\parallel$	\parallel	$\perp$	\perp		

I.28. Операции

log	`\log`	lg	`\lg`	ln	`\ln`
arg	`\arg`	ker	`\ker`	dim	`\dim`
hom	`\hom`	deg	`\deg`	exp	`\exp`
sin	`\sin`	arcsin	`\arcsin`	cos	`\cos`
arccos	`\arccos`	tan	`\tan`	arctan	`\arctan`
cot	`\cot`	sec	`\sec`	csc	`\csc`
sinh	`\sinh`	cosh	`\cosh`	tanh	`\tanh`
coth	`\coth`				

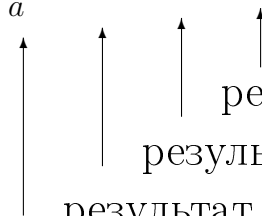
I.29. Операции с пределами

$\sum$	`\sum`	$\prod$	`\prod`	$\bigcup$	`\bigcup`
$\bigcap$	`\bigcap`	$\coprod$	`\coprod`	$\bigoplus$	`\bigoplus`
$\bigotimes$	`\bigotimes`	$\bigodot$	`\bigodot`	$\bigvee$	`\bigvee`
$\bigwedge$	`\bigwedge`	$\biguplus$	`\biguplus`	$\bigsqcup$	`\bigsqcup`
$\lim$	`\lim`	$\limsup$	`\limsup`	$\liminf$	`\liminf`
$\max$	`\max`	$\min$	`\min`	$\sup$	`\sup`
$\inf$	`\inf`	$\det$	`\det`	$\Pr$	`\Pr`
$\gcd$	`\gcd`	$\int$	`\int`		

I.29. Операции с пределами

Обратите внимание на разницу между знаками

$\int_a^b \quad \int\limits_a^b \quad \displaystyle\int_a^b \quad \displaystyle\int\limits_a^b$



результат компиляции выражения `\int_{a}^{b}`

результат компиляции выражения `\int\limits_{a}^{b}`

результат компиляции выражения `\displaystyle\int_{a}^{b}`

результат компиляции для `\displaystyle\int\limits_{a}^{b}`

В «выключных» формулах (переход в которые и выход из которых осуществляется с помощью знака `$$`) режим `\displaystyle` выставлен «по умолчанию».

$\int f(x)g(x)\,dx$ — результат компиляции кода `\int f(x)g(x)\,dx` — здесь выражение «`\,`» обеспечивает вставку «тонкого пробельчика» (шпации), отделяющего `dx` от `f(x)g(x)`.

I.30. Стрелки

$\Longrightarrow$	\Longrightarrow	$\nwarrow$	\nwarrow	$\nearrow$	\nearrow
$\Longleftarrow$	\Longleftarrow	$\uparrow$	\uparrow	$\downarrow$	\downarrow
$\longleftarrow$	\longleftarrow	$\updownarrow$	\updownarrow	$\gets$	\gets
$\longrightarrow$	\longrightarrow	$\searrow$	\searrow	$\swarrow$	\swarrow
$\longleftrightarrow$	\longleftrightarrow	$\leftrightarrow$	\leftrightarrow	$\to$	\to
$\Longleftrightarrow$	\Longleftrightarrow	$\mapsto$	\mapsto	$\longmapsto$	\longmapsto
$\leftharpoondown$	\leftharpoondown	$\Uparrow$	\Uparrow	$\Downarrow$	\Downarrow
$\rightharpoondown$	\rightharpoondown	$\Updownarrow$	\Updownarrow	$\Rightarrow$	\Rightarrow
$\leftharpoonup$	\leftharpoonup	$\Leftrightarrow$	\Leftrightarrow	$\Leftarrow$	\Leftarrow
$\rightharpoonup$	\rightharpoonup	$\hookleftarrow$	\hookleftarrow		
$\rightleftharpoons$	\rightleftharpoons	$\hookrightarrow$	\hookrightarrow	$\leadsto$	\leadsto

I.31. Разные символы

∂	\partial	$\triangle$	\triangle	$\angle$	\angle
∞	\infty	$\forall$	\forall	$\exists$	\exists
$\emptyset$	\emptyset	$\neg$	\neg	$\aleph$	\aleph
$\prime$	\prime	$\hbar$	\hbar	∇	\nabla
$\imath$	\imath	$\jmath$	\jmath	ℓ	\ell
$\surd$	\surd	$\flat$	\flat	$\sharp$	\sharp
$\natural$	\natural	$\top$	\top	$\bot$	\bot
$\wp$	\wp	$\Re$	\Re	$\Im$	\Im
$\backslash$	\backslash	$\|$	|	$\spadesuit$	\spadesuit
$\clubsuit$	\clubsuit	$\diamondsuit$	\diamondsuit	$\heartsuit$	\heartsuit
$\mho$	\mho	$\Box$	\Box	$\Diamond$	\Diamond
$\dagger$	\dag	$\S$	\S	$\copyright$	\copyright
$\ddagger$	\ddag	$\P$	\P	$\pounds$	\pounds

I.32. Некоторые сокращения

$*$	*	\ast	$\neq$	\ne	\neq
$\neq$	\ne	\neq			
$\leq$	\le	\leq	$\geq$	\ge	\geq
$[$	[	\lbrack	$]$	]	\rbrack
$\{$	\{	\lbrace	$\{$	\{	\rbrace
$\rightarrow$	\to	\rightarrow	$\leftarrow$	\gets	\leftarrow
$\ni$	\ni	\owns	$\|$	\Vert	\\|
$\wedge$	\wedge	\land	$\vee$	\vee	\lor
$\neg$	\neg	\lnot			

I.33. Знаки над буквами и строками

`\acute` — $\acute{x}, \acute{y}, \acute{z}, \ldots$ (только математическая мода)

`\bar` — $\bar{x}, \bar{y}, \bar{z}, \ldots$ (только математическая мода)

`\breve` — $\breve{x}, \breve{y}, \breve{z}, \ldots$ (только математическая мода)

`\check` — $\check{x}, \check{y}, \check{z}, \ldots$ (только математическая мода)

`\grave` — $\grave{x}, \grave{y}, \grave{z}, \ldots$ (только математическая мода)

`\ddot` — $\ddot{x}, \ddot{y}, \ddot{z}, \ldots$ (только математическая мода)

`\dot` — $\dot{x}, \dot{y}, \dot{z}, \ldots$ (только математическая мода)

`\hat{символ}` — $\hat{x}, \hat{y}, \hat{z}, \ldots$ (только математическая мода)

`\overline{строка}` — $\overline{x}, \overline{xyz}$ (только математическая мода)

`\tilde` — $\tilde{x}, \tilde{y}, \tilde{z}, \ldots$ (только математическая мода)

`\underline` — $\underline{x}, \underline{xyz}$, $\underline{\text{текст}}$

`\vec` — $\vec{x}, \vec{y}, \vec{z}, \ldots$ (см. `\vectr`, только математическая мода)

`\widehat{символ}` — $\widehat{xyz}, \widehat{xy}, \widehat{z}, \ldots$ (только математическая мода)

I.33. Знаки над буквами и строками

В математической моде знак над буквой можно сделать с помощью `\stackrel`, например, `$\stackrel{x}{a}$` приводит к такому результату компиляции: $\stackrel{x}{a}$.

Аналогом команды `\acute` для текстовой моды является `\'{...}`, а для команды `\ddot` — команда `\"{...}`.

Например, ударе́ние ставится так: `удар\'{е}ние`,
а слово «ёжик» можно получить так: `«\¨{е}жик»` (при использовании кодировки UTF-8 необходимости в последнем ухищрении нет).

I.34. Скобки под и над строками

В математической моде можно поставить скобки под строкой и над строкой.

Например, $x+\underbrace{y+z}_{\alpha\cdot\beta}+t$ является результатом компиляции кода:

```
\mbox{$x+\underbrace{y+z}_{\alpha\cdot\beta}+t$}
```

$x+\overbrace{y+z}^{\alpha\cdot\beta}+t$ является результатом компиляции кода:

```
\mbox{$x+\overbrace{y+z}^{\alpha\cdot\beta}+t$}
```

I.35. Единицы длины

В TEX и LATEXприменяются следующие единицы длины:

`pt` — пункт$\approx 0,35$ мм;

`pc` — пика$= 12$ pt;

`mm` — миллиметр;

`cm` — сантиметр;

`in` — дюйм (inch)$= 25,4$ мм;

`dd` — пункт Дидо$= 1,07$ pt;

`cc` — цицеро$= 12$ dd;

`em` — ширина буквы M в текущем шрифте;

`ex` — высота буквы x в текущем шрифте.

Для того, чтобы установить стандартную единицу длины для команд, использующих безразмерные величины длины, применяется команда `\unitlength{величина длины}`.

I.36. Пробелы

Код	Результат компиляции	Символ «~» означает «неразрывный пробел».
`a\qquad b`	a b	В математической моде пробелы между символами расставляются автоматически, но данные команды позволяют корректировать эти пробелы.
`a\quad b`	a b	
`a\ b`	a b	
`a\;b`	a b	
`a\:b`	a b	
`a\,b`	a b	
`a b`	a b	Кратные пробелы (двойной, тройной и др.) в файле `*.tex` при компиляции игнорируются.
`a  b`	a b	
`a   b`	a b	
`a    b`	a b	
`a\!b`	*ab*	Отрицательный тонкий пробел, используется в математической моде

I.37. Команды, определенные в texsample.tex

Окружения типа «теорема»: нами введены окружения

\begin{thm} ... \end{thm} — «теорема»,

\begin{lmm}...\end{lmm} — «лемма»,

\begin{defnt}...\end{defnt} — «определение»,

\begin{sogl}...\end{sogl} — «соглашение»,

\begin{law} ... \end{law} — «закон»,

\begin{regul} ... \end{regul} — «правило»,

\begin{prim}...\end{prim} — «пример»,

\begin{zad}...\end{zad} — «задача».

\PdfSection, \PdfSubSection, \PdfSubSubSection, \PrimCom, \ThmEnvCom, \PictText, \TextPict, \setka,

\bigref — печатает номер ссылки с номером страницы.

\mathref — печатает номер математической формулы с номером страницы.

I.37. Команды, определенные в texsample.tex

Эти команды работают в математической моде.

\Arg — Arg
\tr — tr
\Rg — Rg
\Ker — Ker
\spec — spec
\diag — diag
\prc — пр
\diez — #
\supp — supp
\Irr — Irr

\vectrup — $\overrightarrow{\mathbf{x}}', \overrightarrow{\mathbf{y}}', \overrightarrow{\mathbf{z}}', \ldots$
\nablavec — $\overrightarrow{\bigtriangledown}$
\nablacoord — $\bigtriangledown_x, \bigtriangledown_y, \bigtriangledown_z, \ldots$
\vectr — $\overrightarrow{\mathbf{x}}, \overrightarrow{\mathbf{y}}, \overrightarrow{\mathbf{z}}, \ldots$ (см.\vec)
\diam — diam
\grad — grad
\rot — rot
\divrg — div

II. Алфавитный список команд LaTeXи введённых в texsample

В отдельный раздел вынесены команды введённые в testsample.tex.

A

`\addcontentsline{toc}{section}{О чем-то великом}` — команда вставки нового элемента «О чем-то великом» в оглавление, точнее в файл с расширением `toc`, содержащий список названий разделов. Если команда имеет вид
`\addcontentsline{toc}{subsection}{О чем-то менее великом}`,
то раздел «О чем-то менее великом» в оглавлении будет представлен как подраздел, т.е. раздел уровня `subsection`. Уровень раздела будет еще ниже — `subsubsection`,— если команда имеет вид `\addcontentsline{toc}{subsubsection}{О скромной нетленке}`.

`\addtocounter` — команда добавления числа к счетчику. Например, команда `\addtocounter{thm}{-2}` означает добавление к счетчику `thm` числа -2.

`\addtolength{команда-длина}{добавляемое значение длины}` — команда прибавления к текущему значению команды-длины добавляемого значения длины.

Например, если была определена новая длина командой `\newlength{\MyLength}`, и командой `\setlength{\MyLength}{12mm}` команде `\MyLength` было присвоено значение `12 mm`, то после `\addtolength{\MyLength}{7mm}` команда `\MyLength` станет равносильна `19 mm`.

`\advance\`*идентификатор_регистра* `by` n — команда, в результате применения которой значение регистра с именем `\`*идентификатор_регистра* увеличится на число n (или уменьшится при $n < 0$).

Специальное слово `by` является необязательным.

Например, выполнение кода

```
\newcount\Cadvance
Начальное значение регистра равно \the\Cadvance,
потом \advance\Cadvance 3  \the\Cadvance,
а теперь \advance\Cadvance -5  $(\the\Cadvance)$
```

приведёт к следующему результату:

Начальное значение регистра равно 0, потом 3, а теперь (-2).

`aim` — имя окружения для описания цели, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании **`\newtheorem`**.

Как для любого другого окружения, с окружением `aim` связан одноименный счетчик, причем после выполнения `\begin{aim}` значение счетчика `aim` увеличивается на 1.

`aim` — имя окружения для описания цели, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`.

Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#aim# равно~\arabic{aim}.
 \begin{aim}
Установить связь между длинами сторон треугольника
с величинами его углов.
 \end{aim}
Теперь значение счетчика~\verb#aim# равно~\arabic{aim}.
```

дает следующий результат:

Начальное значение счетчика `aim` равно 0.

Цель 1. *Установить связь между длинами сторон треугольника с величинами его углов.*

Теперь значение счетчика `aim` равно 1.

`\alph{имя_счётчика}` — команда для печати значения счётчика строчными греческими буквами.

В результате компиляции кода:

```
Значение сч\symbol{188}тчика section
равно \alph{section}=
\Roman{section}= \roman{section}=
\Alph{section}= \arabic{section}
```

получим

«Значение счётчика section равно b= II= ii= B= 2».

`\Alph{имя_счётчика}` — команда для печати значения счётчика заглавными греческими буквами.

В результате компиляции кода:

```
Значение сч\symbol{188}тчика section
равно \Alph{section}=
\Roman{section}= \roman{section}=
\alph{section}= \arabic{section}
```

получим

«Значение счётчика section равно B= II= ii= b= 2».

`\arabic{имя_счётчика}` — команда для печати значения счётчика арабскими цифрами.

В результате компиляции кода:

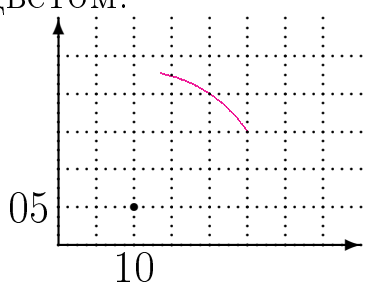
```
Значение сч\symbol{188}тчика section
равно \arabic{section}=
\Roman{section}= \roman{section}=
\Alph{section}= \alph{section}
```

получим

«Значение счётчика section равно 2= II= ii= B= b».

`\arc(dx,dy){a}` изображает дугу окружности. А именно, команда `\put(x,y){\arc(dx,dy){a}}` изображает дугу окружности с центром в точке с координатами `(x,y)`, начинающаяся в точке с координатами `(x+dx,y+dy)`, составляющая `a` градусов (положительное направление — против часовой стрелки).

Например, команда `\put(10,05){\arc(15,10){45}}` приводит к изображению дуги, выделенной на рисунке пурпурным (magenta) цветом:



`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

Каждому из аргументов l, c, r команды `\begin{array}{cl|rcl||r}` соответствует столбец в таблице.

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

Число столбцов равно количеству аргументов команды `\begin{array}{cl|rcl||r}` без учета скобок «|».

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

В строке символы из разных столбцов отделяются друг от друга символом «`&`».

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

Переход к следующей строке осуществляется командой «\\».

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

Аргументы l, c, r команды `\begin{array}{cl|rcl||r}` означают характер выравнивания текста в соответствующем столбце:

l — по левому краю (left),

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

Аргументы l, c, r команды `\begin{array}{cl|rcl||r}` означают характер выравнивания текста в соответствующем столбце:

l — по левому краю (left),

c — по центру столбца (center),

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

Аргументы l, c, r команды `\begin{array}{cl|rcl||r}` означают характер выравнивания текста в соответствующем столбце:

l — по левому краю (left),

c — по центру столбца (center),

r — по правому краю (right).

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

Скобка `|`, разделяющая некоторые аргументы команды `\begin{array}{cl|rcl||r}`, означает проведение вертикальной черты между соответствующими столбцами.

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

Команда `\hline` приводит к проведению горизонтальной черты, разделяющей строки.

`array` — **окружение** для создания таблиц в математической моде, с синтаксисом, аналогичным окружению `tabular`. Например,

$-c$	$-l$	$-r$	$-c$	$-l$	$-r$
c	l	r	c	l	r
$c+$	$l+$	$r+$	$c+$	$l+$	$r+$
1	2	3	4	5	6

есть результат компиляции следующего фрагмента текста

```
\begin{array}{cl|rcl||r}
 -c & -l & -r & -c & -l & -r\\
   \hline
  c &  l &  r &  c &  l &  r\\
 c+ & l+ & r+ & c+ & l+ & r+\\
   \cline{3-5}
  1 &  2 &  3 &  4 &  5 &  6\\
   \cline{6-6}
\end{array}
```

Команда `\cline{a-b}` приводит к проведению горизонтальной черты между столбцами с номерами a и b, разделяющей соответствующие строки.

B

`\begin{имя_окружения}` — команда для начала нового **окружения**.

Завершается **окружение** командой `\end{имя_окружения}`.

Примерами типовых окружения являются `equation`, `center`, `array` `pmatrix` `vmatrix`.

`\bigcircle` — команда из пакета, определяемого стилевым файлом `curves.sty`. Является расширением команды `\circle`, снимающим ограничения на размеры рисуемой окружности.

`\bigskip` — команда для увеличения расстояния между абзацем, в котором встретилась эта команда, и предыдущим абзацем.

Расстояние увеличивается на несколько большую величину, чем при использовании команды `\medskip`.

Здесь была вставлена команда `\medskip`.

Здесь была вставлена команда `\bigskip`.

Вставка заранее определённого расстояния между абзацами осуществляется командой `\vspace`. Например, перед этим абзацем была вставлена команда `\vspace{3ex}`.

`black` — чёрный цвет. Используется, например, в составе команды `{\color{black}ч\symbol{188}рный цвет}`.

`blue` — синий цвет. Используется, например, в составе команды `{\color{blue}синий цвет}`.

C

`center` — окружение, обеспечивающее центрирование всех строк обзаца или нескольких абзацев, охватываемых данным окружением, т.е. расположенных между. `\begin{center}` и `\end{center}`.

В результате компиляции кода:

```
«\begin{center}
Абзац некоторого текста. Строки выравниваются по центру.
Абзац может содержать формулы и рисунки,
например, \mbox{$9x^{2} - 16y^{2} = 25$}.
  Абзацев может быть несколько.
\end{center}»
```

Абзац некоторого текста. Строки выравниваются по центру. Абзац может содержать формулы и рисунки, например, $9x^2 - 16y^2 = 25$. Абзацев может быть несколько.

`\centerline{текстовая строка}` — команда центрирования одной строки.

В результате компиляции кода

```
«Текст абзаца, заканчивающийся в текстовой моде, занимающий,
быть может, несколько строк и завершающийся
\symbol{92}newline.\newline
\centerline{А это строка, напечатанная по центру.}
Продолжение текста.»
```

получаем:

Текст абзаца, заканчивающийся в текстовой моде, занимающий, быть может, несколько строк и завершающийся \newline.

А это строка, напечатанная по центру.

Продолжение текста.

`\centerline{текстовая строка}` — команда центрирования одной строки.

Завешать её действие применением **\newline** не обязательно:

```
«Текст абзаца, заканчивающийся в текстовой моде, занимающий,
быть может, несколько строк и завершающийся
\symbol{92}newline.\newline
\centerline{А это строка, напечатанная по центру.}
\symbol{92}newline
Продолжение текста.»
```

после компиляции даёт:

Текст абзаца, заканчивающийся в текстовой моде, занимающий, быть может, несколько строк и завершающийся **\newline**.

А это строка, напечатанная по центру.

Продолжение текста.

`\centerline{текстовая строка}` — команда центрирования одной строки.

Эта команда должна начинать абзац или предваряться **\newline**, иначе результат компиляции кода

```
«Текст абзаца, заканчивающийся в текстовой моде, занимающий,
быть может, несколько строк.
\centerline{А это строка, напечатанная по центру.}
Продолжение текста.»
```

будет такой:

Текст абзаца, заканчивающийся в текстовой моде, занимающий, быть может, несколько строк.

А это строка, напечатанна

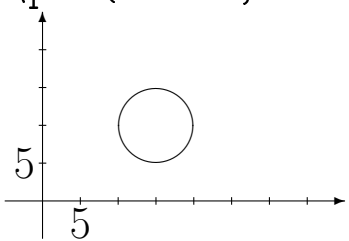
Продолжение текста.

`\cline{`m`-`n`}` — команда проведения горизонтальной линии в таблице определённой с помощью **окружений** `array`, `tabular`, через столбцы с номерами $m, m+1, \ldots, n-1, n$.

`\circle{D}` — команда рисования окружности диаметра D. Используется внутри окружения

```
\begin{picture}(???,???)
\end{picture}
```

Например, результат действия команды `\put(15.00,10.00){\circle{20}}` будет таким:



Имеются некоторые ограничения на возможные значения диаметра. Эти ограничения снимаются при использовании команды `\bigcircle` из пакета, определяемого стилевым файлом `curves.sty`.

`\color{цвет}` — команда выбора цвета объектов.

В цветовой модели `named` доступны следующие цвета:

```
\color{black}black, \color{red}red, \color{magenta}magenta,
\color{cyan}cyan, \color{blue}blue, \color{green}green,
\color{yellow}yellow, \color{white}white.
```

Если `{\color{magenta}` изменение цвета осуществляется внутри группы, то`}` по завершении группы цвет восстанавливается.

`\ComOverlayA{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода фрагментов текста при определении новых команд, введённая в `texsample.tex`. См. `\ComOverlayB`.

Здесь n номер аргумента команды, определяемой с помощью `\def`, `\newcommand`, `\renewcommand`.

n_i — целое число, при `#`n`>`n_i выполняется код σ_i.

```
\def\Xyz#1{Например, \ComOverlayA{#1}
           {0}{печатаем }
           {2}{текст }
           {1}{по очереди.}
           {2}{}}
```

Здесь $n = 1$, $n_1 = 0$, $\sigma_1 =$«печатаем »,
$n_2 = 2$, $\sigma_1 =$«текст »,
$n_3 = 1$, $\sigma_1 =$«по очереди »,
$n_4 = 2$, $\sigma_1 =$«».

`\ComOverlayA{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода фрагментов текста при определении новых команд.

Здесь n номер аргумента команды, определяемой с помощью `\def`, `\newcommand`, `\renewcommand`.

n_i — целое число, при `#`n`>`n_i выполняется код σ_i.

```
\def\Xyz#1{Например, \ComOverlayA{#1}
           {0}{печатаем }
           {2}{текст }
           {1}{по очереди.}
           {2}{}}
\Xyz{0}
```

При компиляции этого кода получим:

«Например, »

`\ComOverlayA{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода фрагментов текста при определении новых команд.

Здесь n номер аргумента команды, определяемой с помощью `\def`, `\newcommand`, `\renewcommand`.

n_i — целое число, при `#`n`>`n_i выполняется код σ_i.

```
\def\Xyz#1{Например, \ComOverlayA{#1}
           {0}{печатаем }
           {2}{текст }
           {1}{по очереди.}
           {2}{}}
\Xyz{1}
```

При компиляции этого кода получим:

«Например, печатаем »

`\ComOverlayA{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода фрагментов текста при определении новых команд.

Здесь n номер аргумента команды, определяемой с помощью `\def`, `\newcommand`, `\renewcommand`.

n_i — целое число, при `#`n`>`n_i выполняется код σ_i.

```
\def\Xyz#1{Например, \ComOverlayA{#1}
           {0}{печатаем }
           {2}{текст }
           {1}{по очереди.}
           {2}{}}
\Xyz{2}
```

При компиляции этого кода получим:

«Например, печатаем текст по очереди.»

`\ComOverlayA{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода фрагментов текста при определении новых команд.

Здесь n номер аргумента команды, определяемой с помощью `\def`, `\newcommand`, `\renewcommand`.

n_i — целое число, при `#`n`>`n_i выполняется код σ_i.

```
\def\Xyz#1{Например, \ComOverlayA{#1}
           {0}{печатаем }
           {2}{текст }
           {1}{по очереди.}
           {2}{}}
\Xyz{3}
```

При компиляции этого кода получим:

«Например, печатаем текст по очереди.»

При значениях аргумента, бóльших 2, печатаются все σ_i.

`\ComOverlayA{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода фрагментов текста при определении новых команд.

Здесь n номер аргумента команды, определяемой с помощью `\def`, `\newcommand`, `\renewcommand`.

n_i — целое число, при `#`n`>`n_i выполняется код σ_i.

```
\def\Xyz#1{Например, \ComOverlayA{#1}
           {0}{печатаем }
           {2}{текст }
           {1}{по очереди.}
           {2}{}}
\Xyz{4}
```

При компиляции этого кода получим:

«Например, печатаем текст по очереди.»

При значениях аргумента, бóльших 2, печатаются все σ_i.

`\ComOverlayA{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода фрагментов текста при определении новых команд.

Здесь n номер аргумента команды, определяемой с помощью `\def`, `\newcommand`, `\renewcommand`.

n_i — целое число, при `#`n`>`n_i выполняется код σ_i.

```
\def\Xyz#1{Например, \ComOverlayA{#1}
           {0}{печатаем }
           {2}{текст }
           {1}{по очереди.}
           {2}{}}
\Xyz{5}
```

При компиляции этого кода получим:

«Например, печатаем текст по очереди.»

При значениях аргумента, бóльших 2, печатаются все σ_i.

`\ComOverlayB{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода строк текста при определении новых команд, введённая в `texsample.tex`. См. `\ComOverlayA`.

Здесь n номер аргумента команды, определяемой с помощью `\def`, `\newcommand`, `\renewcommand`.

`\ComOverlayB{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода строк текста при определении новых команд. См. команду `\ComOverlayA`.

n_i — целое число, при `#`n>n_i выполняется код σ_i в отдельной строке; при `#`$n < 0$ код σ_i игнорируется, строка не вставляется.

```
\def\Xyz#1{Например, \ComOverlayB{#1}
           {0}{печатаем }  {2}{текст }
           {1}{по очереди.}  {-1}{}}
```

Здесь $n = 1$, $n_1 = 0$, $\sigma_1 =$«печатаем »,
$n_2 = 2$, $\sigma_1 =$«текст »,
$n_3 = 1$, $\sigma_1 =$«по очереди »,
$n_4 = 2$, $\sigma_1 =$«».

`\ComOverlayB{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода строк текста при определении новых команд. См. команду `\ComOverlayA`.

n_i — целое число, при #n>n_i выполняется код σ_i в отдельной строке; при #$n < 0$ код σ_i игнорируется, строка не вставляется.

```
\def\Xyz#1{Например, \ComOverlayB{#1}
           {0}{печатаем }  {2}{текст }
           {1}{по очереди.}  {-1}{}}    \Xyz{0}  Строка.
```

При компиляции этого кода получим:

Например,

Строка.

`\ComOverlayB{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода строк текста при определении новых команд. См. команду `\ComOverlayA`.

n_i — целое число, при #n>n_i выполняется код σ_i в отдельной строке; при #$n < 0$ код σ_i игнорируется, строка не вставляется.

```
\def\Xyz#1{Например, \ComOverlayB{#1}
           {0}{печатаем }  {2}{текст }
           {1}{по очереди.}  {-1}{}}    \Xyz{1}  Строка.
```

При компиляции этого кода получим:

Например, печатаем

Строка.

`\ComOverlayB{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода строк текста при определении новых команд. См. команду `\ComOverlayA`.

n_i — целое число, при #n>n_i выполняется код σ_i в отдельной строке; при #$n < 0$ код σ_i игнорируется, строка не вставляется.

```
\def\Xyz#1{Например, \ComOverlayB{#1}
           {0}{печатаем }  {2}{текст }
           {1}{по очереди.}  {-1}{}}    \Xyz{2}  Строка.
```

При компиляции этого кода получим:

Например, печатаем

по очереди.

Строка.

`\ComOverlayB{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода строк текста при определении новых команд. См. команду `\ComOverlayA`.

n_i — целое число, при #n>n_i выполняется код σ_i в отдельной строке; при #$n < 0$ код σ_i игнорируется, строка не вставляется.

```
\def\Xyz#1{Например, \ComOverlayB{#1}
           {0}{печатаем }  {2}{текст }
           {1}{по очереди.}  {-1}{}    \Xyz{3}  Строка.
```

При компиляции этого кода получим:

Например, печатаем
текст
по очереди.
Строка.

`\ComOverlayB{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода строк текста при определении новых команд. См. команду `\ComOverlayA`.

n_i — целое число, при `#`n>n_i выполняется код σ_i в отдельной строке; при `#`$n < 0$ код σ_i игнорируется, строка не вставляется.

```
\def\Xyz#1{Например, \ComOverlayB{#1}
           {0}{печатаем }  {2}{текст }
           {1}{по очереди.}  {2}{}}    \Xyz{3}  Строка.
```

При компиляции этого кода получим:

Например, печатаем
текст
по очереди.

Строка.

`\ComOverlayB{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — команда для обеспечения постепенного вывода строк текста при определении новых команд. См. команду `\ComOverlayA`.

n_i — целое число, при `#`n>n_i выполняется код σ_i в отдельной строке; при `#`$n < 0$ код σ_i игнорируется, строка не вставляется.

```
\def\Xyz#1{Например, \ComOverlayB{#1}
           {0}{печатаем }  {2}{текст }
           {1}{по очереди.}  {2}{}}    \Xyz{3}
```

При компиляции этого кода получим:

Например, печатаем
текст
по очереди.

При значениях аргумента, бóльших 2, печатаются все σ_i.

`\ComOverlayD{#`n`}{`n_1`}{`σ_1`}{`n_2`}{`σ_2`}{`n_3`}{`σ_3`}{`n_4`}{`σ_4`}` — изменённый вариант команды `\ComOverlayA` для обеспечения постепенного вывода фрагментов текста при определении новых команд, введённая в `texsample.tex`.

n_i — целое число, при `#`n`>`n_i выполняется код σ_i.

Отличие от `\ComOverlayA`: при $n \leqslant n_i$ место под σ_i не резервируется.

```
\def\Xyz#1{Например, \ComOverlayD{#1}
{0}{печатаем\quad}{1}{текст\quad}{2}{по\quad}{3}{очереди}.}
```

Код	**Результат компиляции**
`\Xyz{0}`	Например, .
`\Xyz{1}`	Например, печатаем .
`\Xyz{2}`	Например, печатаем текст .
`\Xyz{3}`	Например, печатаем текст по .
`\Xyz{4}`	Например, печатаем текст по очереди.

`\ComOverlayX{#`n`}{`n_1`}{`σ_1`}` — сокращенный вариант команды `\ComOverlayA` для обеспечения постепенного вывода фрагментов текста при определении новых команд, введённая в `texsample.tex`.

Здесь n номер аргумента команды, определяемой с помощью `\def`, `\newcommand`, `\renewcommand`.

n_1 — целое число, при `#`n`>`n_1 выполняется код σ_1.

```
\def\Xyz#1{Печатаем \ComOverlayX{#1}{0}{текст}  по очереди.}
```

Код	**Результат компиляции**
`\Xyz{0}`	Печатаем по очереди.
`\Xyz{1}`	Печатаем текст по очереди.
`\Xyz{2}`	Печатаем текст по очереди.

`\ComSystCoord{`a`}{`b`}{`m_-`}{`m_+`}{`n_-`}{`n_+`}{`x`}{`y`}{K}` — команда для изображения системы координат, введённая в `texsample.tex`, используется внутри окружения `\picture`, в частности, в команде `\TextPict` или `\PictText`:

a — расстояние между метками на оси абсцисс (измеряемое в `\unitlength`, т.е. a — это безразмерное число);

b — расстояние между метками на оси ординат (тоже измеряемое в `\unitlength`);

m_- — число отметок на оси абсцисс слева от начала координат;

m_+ — число отметок на оси абсцисс справа от начала координат;

n_- — число отметок на оси ординат ниже начала координат;

n_- — число отметок на оси ординат выше начала координат;

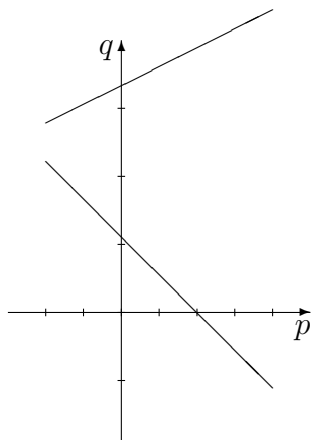
x — обозначение оси абсцисс;

y — обозначение оси ординат;

K — система команд рисования фигур в данной системе координат.

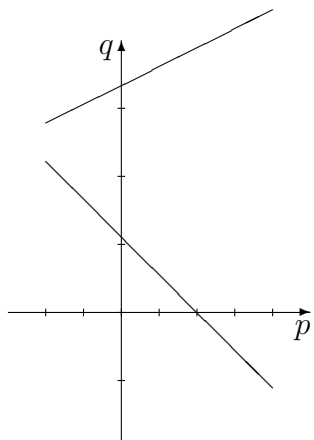
\ComSystCoord{a}{b}{m_-}{m_+}{n_-}{n_+}{x}{y}{K} — команда для изображения системы координат, введённая в texsample.tex, используется внутри окружения \picture, в частности, в команде \TextPict или \PictText.

```
\TextPict{40}{54}{15}{18}{%
  \ComSystCoord{5}{9}{3}{5}{2}{4}{p}{q}{%
      \put(-10.00,025.00){\line( 2, 1){030.00}}
      \put(-10.00,020.00){\line( 1,-1){030.00}}
      }
  }{Данный код TeX.
    }{}{}{}
```



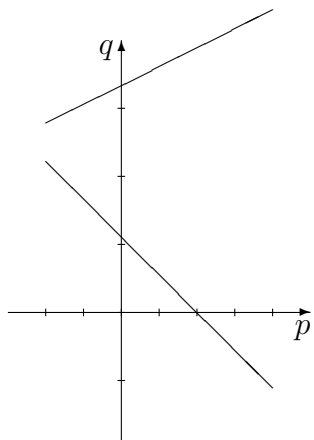
\ComSystCoord{a}{b}{m_-}{m_+}{n_-}{n_+}{x}{y}{K} — команда для изображения системы координат, введённая в texsample.tex, используется внутри окружения \picture, в частности, в команде \TextPict или \PictText.

```
\TextPict{40}{54}{15}{18}{%
  \ComSystCoord{5}{9}{3}{5}{2}{4}{p}{q}{%
      \put(-10.00,025.00){\line( 2, 1){030.00}}
      \put(-10.00,020.00){\line( 1,-1){030.00}}
      }
  }{Данный код TeX.
    }{}{}{}
```



\ComSystCoord{a}{b}{m_-}{m_+}{n_-}{n_+}{x}{y}{K} — команда для изображения системы координат, введённая в texsample.tex, используется внутри окружения \picture, в частности, в команде \TextPict или \PictText.

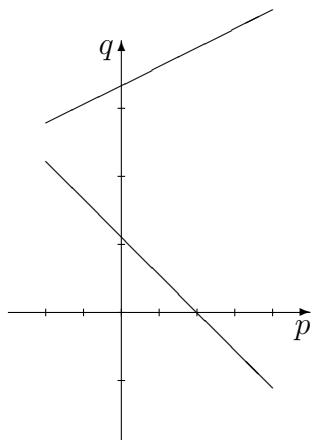
```
\TextPict{40}{54}{15}{18}{%
  \ComSystCoord{5}{9}{3}{5}{2}{4}{p}{q}{%
      \put(-10.00,025.00){\line( 2, 1){030.00}}
      \put(-10.00,020.00){\line( 1,-1){030.00}}
      }
  }{Данный код TeX.
    }{}{}{}
```



$5 \cdot (3 + 5) = 40$

\ComSystCoord{a}{b}{m_-}{m_+}{n_-}{n_+}{x}{y}{K} — команда для изображения системы координат, введённая в texsample.tex, используется внутри окружения \picture, в частности, в команде \TextPict или \PictText.

```
\TextPict{40}{54}{15}{18}{%
  \ComSystCoord{5}{9}{3}{5}{2}{4}{p}{q}{%
      \put(-10.00,025.00){\line( 2, 1){030.00}}
      \put(-10.00,020.00){\line( 1,-1){030.00}}
      }
  }{Данный код TeX.
    }{}{}{}
```

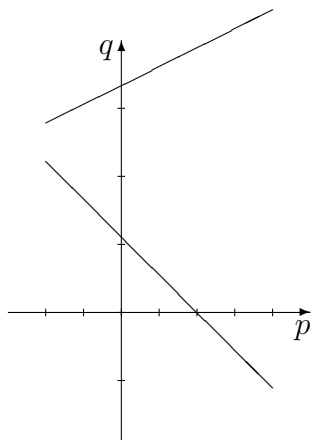


$5 \cdot (3 + 5) = 40$

$5 \cdot 3 = 15$

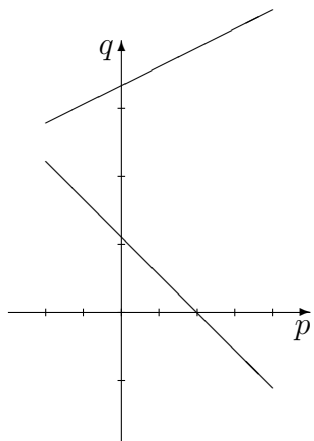
\ComSystCoord{a}{b}{m_-}{m_+}{n_-}{n_+}{x}{y}{K} — команда для изображения системы координат, введённая в texsample.tex, используется внутри окружения \picture, в частности, в команде \TextPict или \PictText.

```
\TextPict{40}{54}{15}{18}{%
  \ComSystCoord{5}{9}{3}{5}{2}{4}{p}{q}{%
      \put(-10.00,025.00){\line( 2, 1){030.00}}
      \put(-10.00,020.00){\line( 1,-1){030.00}}
      }
  }{Данный код TeX.
    }{}{}{}
```



\ComSystCoord{a}{b}{m_-}{m_+}{n_-}{n_+}{x}{y}{K} — команда для изображения системы координат, введённая в texsample.tex, используется внутри окружения \picture, в частности, в команде \TextPict или \PictText.

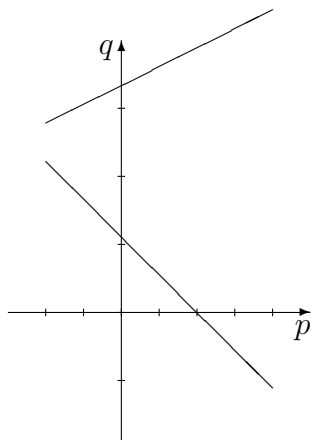
```
\TextPict{40}{54}{15}{18}{%
  \ComSystCoord{5}{9}{3}{5}{2}{4}{p}{q}{%
      \put(-10.00,025.00){\line( 2, 1){030.00}}
      \put(-10.00,020.00){\line( 1,-1){030.00}}
      }
  }{Данный код TeX.
    }{}{}{}
```



$9 \cdot (2 + 4) = 54$

\ComSystCoord{a}{b}{m_-}{m_+}{n_-}{n_+}{x}{y}{K} — команда для изображения системы координат, введённая в texsample.tex, используется внутри окружения \picture, в частности, в команде \TextPict или \PictText.

```
\TextPict{40}{54}{15}{18}{%
  \ComSystCoord{5}{9}{3}{5}{2}{4}{p}{q}{%
      \put(-10.00,025.00){\line( 2, 1){030.00}}
      \put(-10.00,020.00){\line( 1,-1){030.00}}
      }
  }{Данный код TeX.
    }{}{}{}
```

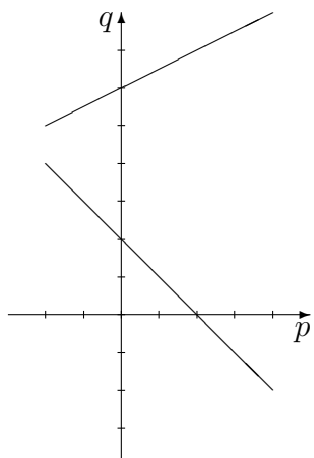


$9 \cdot (2 + 4) = 54$

$9 \cdot 2 = 18$

\ComSystCoord{a}{b}{m_-}{m_+}{n_-}{n_+}{x}{y}{K} — команда для изображения системы координат, введённая в texsample.tex, используется внутри окружения \picture, в частности, в команде \TextPict или \PictText.

```
\TextPict{40}{60}{15}{20}{%
  \ComSystCoord{5}{5}{3}{5}{4}{8}{p}{q}{%
      \put(-10.00,025.00){\line( 2, 1){030.00}}
      \put(-10.00,020.00){\line( 1,-1){030.00}}
      }
  }{Данный код TeX.
    }{}{}{}
```



`cons` — имя окружения для оформления следствий, определенное в файле `texsample.tex`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании **\newtheorem**.

Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`. Как для любого другого окружения, с окружением `cons` связан одноименный счетчик, причем после выполнения `\begin{cons}` значение счетчика `cons` увеличивается на 1.

`cons` — имя окружения для оформления следствий, определенное в файле `texsample.tex`.

Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#cons# равно~\arabic{cons}.
 \begin{cons}
Треугольник является равнобедренным тогда и только тогда
когда два какие-либо его угла равны по величине.
 \end{cons}
Теперь значение счетчика~\verb#cons# равно~\arabic{cons}.
```

дает следующий результат:

Начальное значение счетчика `cons` равно 0.

Следствие 1. *Треугольник является равнобедренным тогда и только тогда когда два какие-либо его угла равны по величине.*

Теперь значение счетчика `cons` равно 1.

`\csname` — часть команды `\csname имя_команды \endcsname`, где `имя_команды` не содержит символа `\`. Синоним `\имя_команды` при определении команды с этим именем с помощью `\def`, а также при выпрлнении команды с этим именем.

Например,

```
\def\csname XYZ\endcsname{...}
```

является синонимом для

```
\def\XYZ{...},
```

а код

```
Текст \csname XYZ\endcsname продолжение текста
```

является синонимом для

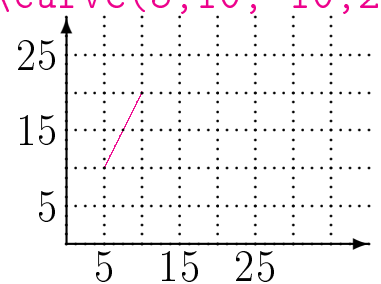
```
Текст \XYZ продолжение текста.
```

`cyan` — голубой цвет. Используется, например, в составе команды `{\color{cyan}голубой цвет}` (в данном случае это не гиперссылка).

`\curve(x1,y1, x2,y2,... xN,yN)` — команда из пакета, подгружаемого стилевым файлом `curves.sty`. В частности, команда `\curve(x1,y1, x2,y2)` рисует отрезок прямой от точки с координатами $(x1, y1)$ до точки с координатами $(x2, y2)$. Команда `\curve(x1,y1, x2,y2, x3,y3)` рисует дугу параболы, проходящую через точки с координатами $(x1, y1)$, $(x2, y2)$, $(x3, y3)$.

На рисунке изображены результаты применения команд:

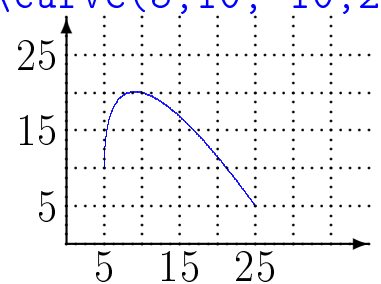
```
\curve(5,10, 10,20)
```



`\curve(x1,y1, x2,y2,... xN,yN)` — команда из пакета, подгружаемого стилевым файлом `curves.sty`. В частности, команда `\curve(x1,y1, x2,y2)` рисует отрезок прямой от точки с координатами $(x1, y1)$ до точки с координатами $(x2, y2)$. Команда `\curve(x1,y1, x2,y2, x3,y3)` рисует дугу параболы, проходящую через точки с координатами $(x1, y1)$, $(x2, y2)$, $(x3, y3)$.

На рисунке изображены результаты применения команд:

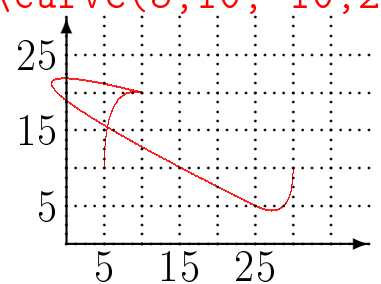
```
\curve(5,10, 10,20, 25,5)
```



`\curve(x1,y1, x2,y2,... xN,yN)` — команда из пакета, подгружаемого стилевым файлом `curves.sty`. В частности, команда `\curve(x1,y1, x2,y2)` рисует отрезок прямой от точки с координатами $(x1, y1)$ до точки с координатами $(x2, y2)$. Команда `\curve(x1,y1, x2,y2, x3,y3)` рисует дугу параболы, проходящую через точки с координатами $(x1, y1)$, $(x2, y2)$, $(x3, y3)$.

На рисунке изображены результаты применения команд:

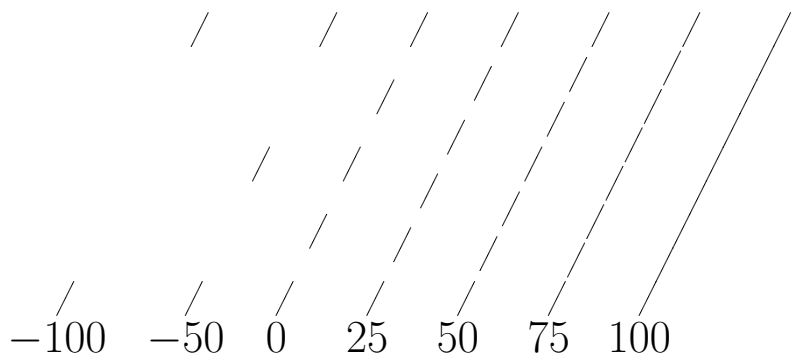
```
\curve(5,10, 10,20, 25,5, 30,10)
```



D

`\dashline[stretch]{length}[IDG]` $(x_1, y_1)\,(x_2, y_2)\ldots(x_n, y_n)$ — команда из пакета `epic` для рисования пунктирных ломаных. Здесь `stretch` — параметр, определяющий расстояние между пунктирными отрезками, это целое число, большее (-100);
`length` — длина отрезка, составляющего пунктирную линию;
`IDG` — положительное число, характеризующее просвет между точками в отрезке, составляющем пунктирную линию;
$(x_1, y_1)\,(x_2, y_2)\ldots(x_n, y_n)$ — координаты точек.

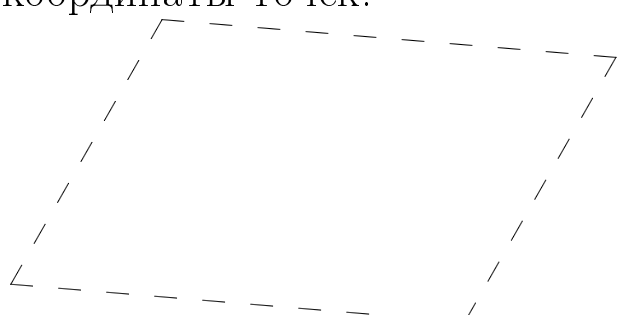
Для разных значений параметра `stretch` приведены результаты применения команды




```
\dashline[stretch]{005.00}(000.00,000.00)(020.00,040.00)
```

`\dashline[stretch]{length}[IDG]` $(x_1, y_1)\,(x_2, y_2)\ldots(x_n, y_n)$ — команда из пакета `epic` для рисования пунктирных ломаных. Здесь `stretch` — параметр, определяющий расстояние между пунктирными отрезками, это целое число, большее (-100);
`length` — длина отрезка, составляющего пунктирную линию;
`IDG` — положительное число, характеризующее просвет между точками в отрезке, составляющем пунктирную линию;
$(x_1, y_1)\,(x_2, y_2)\ldots(x_n, y_n)$ — координаты точек.

Эта команда может использоваться для построения замкнутых ломаных. Вот результат применения команды



```
\dashline{003.00}(000.00,000.00)(020.00,040.00)%
(080.00,035.00)(060.00,-05.00)(000.00,000.00)
```

`\dashline[stretch]{length}[IDG]` $(x_1, y_1)\,(x_2, y_2)\ldots(x_n, y_n)$ — команда из пакета `epic` для рисования пунктирных ломаных. Здесь `stretch` — параметр, определяющий расстояние между пунктирными отрезками, это целое число, большее (-100);
`length` — длина отрезка, составляющего пунктирную линию;
`IDG` — положительное число, характеризующее просвет между точками в отрезке, составляющем пунктирную линию;
$(x_1, y_1)\,(x_2, y_2)\ldots(x_n, y_n)$ — координаты точек.

Для разных значений параметра `EDG`, (просвет между точками) приведены результаты применения команды


```
\dashline{005.00}[EDG](000.00,000.00)(020.00,040.00)
```

`\def\a{текст команды}` — команда для определения макрокоманды `\a`. Здесь под «a» понимается последовательность букв, без цифр, знаков препинания и других символов (например, %, $ и т.п.).

В имени команды лучше использовать только буквы латинского алфавита. См. также `\newcommand`, `\renewcommand`.

Пробелы, следующие непосредственно за командой, компилятор TEXвоспринимает как знак окончания имени команды, т.е. эти пробелы на печати игнорируются. Например,

```
\def\Aa{пере}
\Aa ехал, \Aa шел, \Aa двинул...
```

после компиляции даёт:

переехал, перешел, передвинул...

а не

пере ехал, пере шел, пере двинул...

`\def\a{текст команды}` — команда для определения макрокоманды `\a`. Здесь под «a» понимается последовательность букв, без цифр, знаков препинания и других символов (например, %, $ и т.п.).

Например,

```
\def\USPU{Уральский государственный педуниверситет}
```

после компиляции строки `Наш \USPU \textbf{---} лучший!` выдает следующий результат:

Наш Уральский государственный педуниверситет— лучший!

У команды `\def` могут быть необязательные аргументы `#1`, `#2`, `#3` и т.д. Например, `\def\pqr#1#2{#1 \textbf{---} это #2}`

после компиляции строки

```
\pqr{математика}{жизнь} и \pqr{жизнь}{математика}
```

выдает следующий результат:

математика — это жизнь и жизнь — это математика

`defnt` — имя окружения для оформления определений, определенное в файле `texsample.tex`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании **\newtheorem**.

Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`. Как для любого другого окружения, с окружением `defnt` связан одноименный счетчик, причем после выполнения `\begin{defnt}` значение счетчика `defnt` увеличивается на 1.

`defnt` — имя окружения для оформления определений, определенное в файле `texsample.tex`.

Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#defnt# равно~\arabic{defnt}
 \begin{defnt}
Треугольник называется {\bf равнобедренным}, если
две какие-либо его стороны равны по длине.
 \end{defnt}
Теперь значение счетчика~\verb#defnt# равно~\arabic{defnt}.
```

дает следующий результат:

Начальное значение счетчика `defnt` равно 0.

Определение 1. *Треугольник называется* **равнобедренным**, *если две какие-либо его стороны равны по длине.*

Теперь значение счетчика `defnt` равно 1.

`\displaystyle` — команда переключения в режим печати в «полном режиме».

Например, компиляция фрагмента

```
$\frac{2}{3}$, $\displaystyle\frac{2}{3}$,
$\int_{a}^{b}x\,dx$, $\displaystyle\int\{a}^{b}x\,dx$.
```

приводит к такому результату:

$\frac{2}{3}$, $\displaystyle\frac{2}{3}$, $\int_{a}^{b}x\,dx$, $\displaystyle\int\{a}^{b}x\,dx$.

`\divide\`*идентификатор_регистра* `by` n — команда, в результате применения которой значение регистра с именем `\`*идентификатор_регистра* станет равным целой части от дроби с числителем, равным начальному значению регистра с именем `\`*идентификатор_регистра*, а значенатель равен n.

Специальное слово `by` является необязательным.

Например, выполнение кода

```
\newcount\Cadvance
Начальное значение регистра равно \Cadvance=27
\the\Cadvance,
потом \divide\Cadvance 4  \the\Cadvance,
а теперь \divide\Cadvance -2  $(\the\Cadvance)$
```

приведёт к следующему результату:

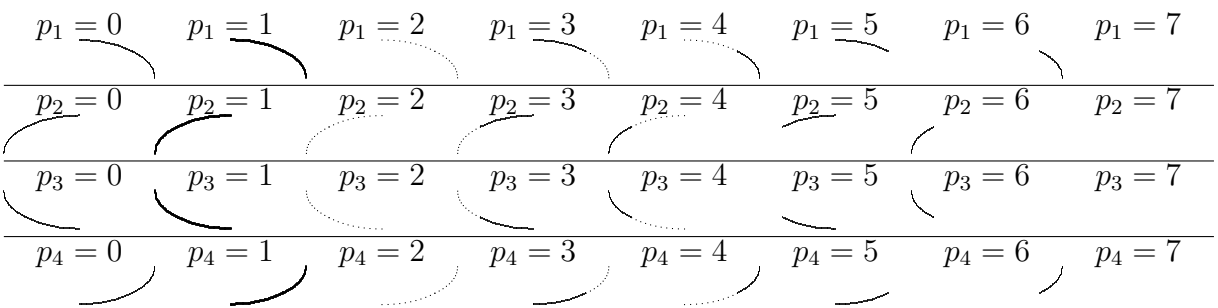
Начальное значение регистра равно 27, потом 6, а теперь (-3).

E

`\ellipsA{`a`}{`b`}{`p_1`}{`p_2`}{`p_3`}{`p_4`}{`n`}{color}` — команда рисования эллипса, где

a, b — длины полуосей, заданные в целых числах, измеряемые в 10^{-2} мм, т.е., например, 3000 — это 30 мм;

$0 \le p_i \le 7$, причём



n — количество точек в соответствующей восьмой части эллипса при $2 \le p_i \le 4$;

color – слово обозначающее цвет тех четвертей эллипса, для которых $p_i = 1$, причём

color $\in$ {black, blue, cyan, green, magenta, red, yellow, white}.

`\else` — команда для реализации альтернативного условия в команде `\ifnum`

`\end{имя_окружения}` — команда для завершения **окружения**.

Начинается **окружение** командой `\begin{имя_окружения}`.

Примерами типовых окружения являются `equation`, `center`, `array`.

`em` — **единица длины,** равная ширине самой широкой буквы: w,щ.

`ex` — **единица длины,** равная высоте буквы x.

Эти единицы длины, естественно, зависят от **типа** и **размера** соответствующего шрифта.

`\endcsname` — часть команды `\csname имя_команды \endcsname`, где `имя_команды` не содержит символа `\`. Синоним `\имя_команды` при определении команды с этим именем с помощью `\def`, а также при выпрлнении команды с этим именем.

Например,

```
\def\csname XYZ\endcsname{...}
```

является синонимом для

```
\def\XYZ{...},
```

а код

```
Текст \csname XYZ\endcsname продолжение текста
```

является синонимом для

```
Текст \XYZ продолжение текста.
```

`equation` — типовое окружение LaTeX для оформления нумерованных выражений.

Как и для других окружений, определен счетчик `equation`, значение которого увеличивается на 1 после выполнения `\begin{equation}`.

`equation` — типовое окружение LaTeX для оформления нумерованных выражений.

Например, рассмотрим фрагмент:

```
Начальное значение сч\"{e}тчика \texttt{equation}
равно \arabic{equation}.
    \begin{equation}
y=x+1.
    \end{equation}
Теперь значение сч\"{e}тчика \texttt{equation}
равно \arabic{equation}.
```

Результат компилирования этого фрагмента:

Начальное значение счётчика `equation` равно 0.

$$y = x + 1. \tag{1}$$

Теперь значение счётчика `equation` равно 1.

`\expandafter\имя_команды` — команда, выполняемая при определении `некоторой_команды` с помощью `\def`, `\newcommand`, `\renewcommand`, предписывающая отложить выполнение команды с именем «`имя_команды`», т.е выполнить команду с именем «`имя_команды`» не на этапе определения `некоторой_команды`, а на этапе выполнения `некоторой_команды`.

Например, при попытке компиляции кода

```
\def\Abc#1#2{                \def\csname #1\endcsname{#2}}
```

будет выдана ошибка, поскольку будет сделана попытка создать команду `\csname` (которая, кстати, уже определена), но после `\csname` компилятор будет ожидать скобку `}`.

`\expandafter\имя_команды` — команда, выполняемая при определении `некоторой_команды` с помощью `\def`, `\newcommand`, `\renewcommand`, предписывающая отложить выполнение команды с именем «`имя_команды`», т.е выполнить команду с именем «`имя_команды`» не на этапе определения `некоторой_команды`, а на этапе выполнения `некоторой_команды`.

Надо написать
`\def\Abc#1#2{\expandafter\def\csname #1\endcsname{#2}}`,
чтобы «внутренняя» команда `\def` не выполнялась на этапе определения `\Abc`. Это действие будет выполнено только, например, при компиляции кода `\Abc{xx}{yyy}`. При этом будет определена команда `\xx`, которая печатает `yyy`.

F

`\fbox{Ф}` — команда, схожая по структуре с командой `\mbox`, отличающаяся тем, что содержимое Ф создаваемого бокса помещается в рамку. Например, результат выполнения команд `\fbox{abcd}`, `\fbox{\fbox{abcd}}`, `\fbox{a\fbox{bc}d}`, `\fbox{a\fbox{b\fbox{c}}d}` имеет вид
abcd, abcd, abcd, abcd.

`\fi` — команда для завершения действия команд `\if...`, т.е. `\ifnum`, `\ifcase` и др.

`\fTwoBlock{...}{...}{...}{...}{...}` — команда, **определённая** в `texsample.tex`, обеспечивающая вывод информации в виде двух горизонтальных блоков одинаковой высоты, (см. `TwoBlock`).

Первый и второй аргументы `#1` и `#2` означают ширину первого и, соответсвенно, второго блока, измеряемую в долях от ширины текста, т.е. от значения переменной `\textwidth`.

Третий аргумент `#3` равен высоте блоков (высота у них будет одинаковой) в тех же единицах.

Значением четвертого `#4` и пятого аргумента `#5` являются содержание первого и, соответственно, второго блока. При этом текст автоматически форматируется по ширине столбца.

`\fTwoBlock{...}{...}{...}{...}{...}` — команда, **определённая** в `texsample.tex`, обеспечивающая вывод информации в виде двух горизонтальных блоков одинаковой высоты, (см. `TwoBlock`).

Компиляция фрагмента

```
\fTwoBlock{0.2}{0.25}{0.1} {Первая колонка текста}
{Вторая колонка текста.}
```

даёт следующий результат:

Первая колонка текста	Вторая колонка текста

`\fTwoBlock{...}{...}{...}{...}{...}` — команда, **определённая** в `texsample.tex`, обеспечивающая вывод информации в виде двух горизонтальных блоков одинаковой высоты, (см. TwoBlock).

Компиляция фрагмента

```
\fTwoBlock{0.4}{0.45}{0.15} {\color{magenta}Первая колонка,
шириной \mbox{\texttt{0.4\symbol{92}textwidth}}, высотой
\mbox{\texttt{0.15\symbol92textwidth}.}
{Вторая колонка, шириной
\mbox{\texttt{0.45\symbol{92}textwidth}},
высотой \mbox{\texttt{0.15\symbol{92}textwidth}}.}}
```

даёт следующий результат:

Первая колонка, шириной 0.4\textwidth, высотой 0.2\textwidth.

Вторая колонка, шириной 0.45\textwidth, высотой 0.2\textwidth.

`\frac` — команда печати обыкновенной дроби. Обычно работает только в математической моде.

Например,
`$\frac{2}{n}$` компилируется как $\frac{2}{n}$.

Для того, чтобы шрифт в числителе и знаменателе имел тот же размер, что и в основном тексте, надо набирать `\displaystyle\frac` вместо `\frac`.

Например после компиляции кода

```
${\displaystyle\frac{x+1}{x^{2}+1}}= \frac{x+1}{x^{2}+1}$
```

получим

$$\frac{x+1}{x^2+1} = \frac{x+1}{x^2+1}$$

G

`\Gamma` — греческая буква Γ.

`\gamma` — греческая буква γ.

Используются только в ***математической моде!***

`gather` — окружение для введения нескольких выключных пронумерованных формул. Например,

$$f(x) = x^2 - 1 \tag{2}$$

$$f(x) = (x - 1)(x + 1) \tag{3}$$

является результатом компиляции следующего фрагмента:

`gather` — окружение для введения нескольких выключных пронумерованных формул. Например,

$$f(x) = x^2 - 1 \tag{2}$$
$$f(x) = (x - 1)(x + 1) \tag{3}$$

является результатом компиляции следующего фрагмента:

```
\begin{gather}
  f(x)=x^{2}-1\mylabel{GatherA}\\
  f(x)=(x-1)(x+1)\mylabel{GatherB}
\end{gather}
```

`gather` — окружение для введения нескольких выключных пронумерованных формул. Например,

$$f(x) = x^2 - 1 \tag{2}$$
$$f(x) = (x - 1)(x + 1) \tag{3}$$

является результатом компиляции следующего фрагмента:

```
\begin{gather}
  f(x)=x^{2}-1\mylabel{GatherA}\\
  f(x)=(x-1)(x+1)\mylabel{GatherB}
\end{gather}
```

Можно осуществить ссылку на **формулу (2)** или на **формулу (3)**.

\ge — математический символ $\ge$.

`\global` — команда, применяемая для того, чтобы распространить на весь документ действие команды, определенной внутри группы `{...}` с помощью `\def`, `\newcommand` и др.

`green` — зелёный цвет. Используется, например, в составе команды `{\color{green}зел\symbol{188}ный цвет}`.

H

`\hbox to` L `{`первая_часть `\hss` вторая_часть`}` — команда для размещения текста таким образом, чтобы последний символ «ь» из фрагмента строки, обозначенного как «вторая часть», находился на расстоянии L (указание единицы измерения для L обязательно) от первого символа «п» из фрагмента строки «первая_часть».

Смещение происходит за счёт «клея», обозначенного как `\hss`.

Например, результат компиляции кода:

```
\hbox to 9cm {Начало \hss и конец фразы}\newline
```

будет таким:

Начало и конец фразы

`\hbox to` L `{`первая_часть `\hss` вторая_часть`}` — команда для размещения текста таким образом, чтобы последний символ «ь» из фрагмента строки, обозначенного как «вторая часть», находился на расстоянии L (указание единицы измерения для L обязательно) от первого символа «п» из фрагмента строки «первая_часть».

Смещение происходит за счёт «клея», обозначенного как `\hss`.

Например, результат компиляции кода:

```
\hbox to 6cm {Начало \hss и конец фразы}\newline
```

будет таким:

Начало и конец фразы

`\hbox to` L `{`первая_часть `\hss` вторая_часть`}` — команда для размещения текста таким образом, чтобы последний символ «ь» из фрагмента строки, обозначенного как «вторая часть», находился на расстоянии L (указание единицы измерения для L обязательно) от первого символа «п» из фрагмента строки «первая_часть».

Смещение происходит за счёт «клея», обозначенного как `\hss`.

Например, результат компиляции кода:

```
\hbox to 5cm {Начало \hss и конец фразы}\newline
```

будет таким:

Начало и конец фразы

`\hbox to` L `{`первая_часть `\hss` вторая_часть`}` — команда для размещения текста таким образом, чтобы последний символ «ь» из фрагмента строки, обозначенного как «вторая часть», находился на расстоянии L (указание единицы измерения для L обязательно) от первого символа «п» из фрагмента строки «первая_часть».

Смещение происходит за счёт «клея», обозначенного как `\hss`.

Например, результат компиляции кода:

```
\hbox to 3cm {Начало \hss и конец фразы}\newline
```

будет таким:

`\hbox to` L `{`первая_часть `\hss` вторая_часть`}` — команда для размещения текста таким образом, чтобы последний символ «ь» из фрагмента строки, обозначенного как «вторая часть», находился на расстоянии L (указание единицы измерения для L обязательно) от первого символа «п» из фрагмента строки «первая_часть».

Смещение происходит за счёт «клея», обозначенного как `\hss`.

Например, результат компиляции кода:

```
\hbox to 0cm {Начало \hss и конец фразы}\newline
```

будет таким:

и конец фразыНачало

`\hbox to` L `{`первая_часть `\hss` вторая_часть`}` — команда для размещения текста таким образом, чтобы последний символ «ь» из фрагмента строки, обозначенного как «вторая часть», находился на расстоянии L (указание единицы измерения для L обязательно) от первого символа «п» из фрагмента строки «первая_часть».

Смещение происходит за счёт «клея», обозначенного как `\hss`.

Например, результат компиляции кода:

```
\hbox to -1cm {Начало \hss и конец фразы}\newline
```

будет таким:

конец фразы Начало

`\hline` — команда проведения горизонтальной линии во всю ширину таблицы, определённой с помощью **окружений** `array`, `tabular`.

`\href{aaa.pdf#BBB}{CCC}` — команда, которая создает гиперссылку в файл `aaa.pdf` на мишень `BBB`. Здесь под `CCC` понимается любой текст. В результате компиляции в выходном pdf-файле текст `CCC` будем «подсвечиваться» голубым (cyan) цветом. При наведении указателя мыши на этот «подсвеченный» текст и нажатия на левую кнопку мыши произойдет переход по гиперссылке на мишень `BBB` в файле `aaa.pdf`

Мишень для `\href` и `\hyperlink` ставится командой `\hypertarget`.

Команды `\hyperlink`, `\href` и `\hypertarget` выполняют указанные действия только при компиляции исходного tex-файла с помощью программы `pdflatex`!

`\hss` — команда вставки «пружинки» («клея»), например, для команды `\hbox`.

`\Huge` — команда переключения размера шрифта на самый большой шрифт.

`\huge` — команда переключения размера шрифта на второй по величине шрифт.

Например, {\Huge Самый большой}, {\huge поменьше}, {\LARGE ещё меньше}, {\Large ещё меньше}, {\large ещё меньше}, {\normalsize ещё меньше}, {\small ещё меньше}, {\tiny совсем мелко}.

`\hyperlink{AAA}{BBB}` — команда, создающая гиперссылку в выходном файле формата pdf . Гиперссылка проводится на мишень с именем, равным первому аргументу команды `\hyperlink{AAA}{BBB}` (в рассматриваемом примере это `AAA`). Второй аргумент — в данном примере это `BBB` — это произвольный текст. После компиляции в выходном pdf-файле этот текст будет «подсвечиваться» красным цветом. При наведении указателя мыши на такой «подсвеченный» текст после нажатия на левую кнопку мыши происходит переход по гиперссылке на мишень с именем `AAA`.

Мишень для `\hyperlink` и `\href` ставится командой `\hypertarget`.

Команды `\hyperlink`, `\href` и `\hypertarget` выполняют указанные действия только при компиляции исходного tex-файла с помощью программы `pdflatex`!

`\hypertarget{L}{C}` — команда, создающая мишень для гиперссылки, определяемой с помощью команд `\hyperlink` и `\href`. Первый аргумент прописывает метку мишени (откуда идет запрос (в рассматриваемом примере это `L`)). Второй аргумент — в данном примере это `C` — задает фрагмент текста в качестве мишени (в данной команде этот фрагмент текста может быть пустым).

Команды `\hyperlink`, `\href` и `\hypertarget` выполняют указанные действия только при компиляции исходного tex-файла с помощью программы `pdflatex`!

I

`\ifcase` — команда выбора из нескольких вариантов:

```
\ifcase A {код, если A=0}
\or{код, если A=1}
\or{код, если A=2}
...
\fi
```

Применяется при определении команд с помощью `def`, `newcommand`, `renewcommand`.

`\ifnum` — команда условия, например:

`\ifnum A<B` *код, если «да»* `\else` *код, если «нет»* `\fi`,

а также

`\ifnum A=B` *код, если «да»* `\else` *код, если «нет»* `\fi`,

и

`\ifnum A>B` *код, если «да»* `\else` *код, если «нет»* `\fi`.

Применяется при определении команд с помощью `def`, `newcommand`, `renewcommand`.

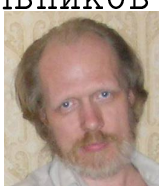
`\includegraphics[scale=m]{filename}` — команда для включения в текст рисунка, содержащегося в файле `filename`.

Возможно использование разных форматов графических файлов, например, pdf и jpg. Необязательный аргумент `scale` означает масштабирующий множитель. Вместо `scale` можно использовать необязательные аргументы `width` и `heigth`, означающие ширину и, соответственно, высоту рисунка. Если исходный размер рисунка был иным, то он будет растянут или сжать до необходимого размера.

`\includegraphics[scale=m]{filename}` — команда для включения в текст рисунка, содержащегося в файле `filename`.

Например, таков результат применения команды

```
фото \includegraphics[scale=0.3]{yury.jpg}
\includegraphics[height=10mm]{yury.pdf} \textbf{--}
Ю.Б.Мельников
```

фото   — Ю.Б.Мельников

Команду `\includegraphics[scale=m]{filename}` можно использовать в качестве задания фигуры Ф в команде `\put`, например,

```
\put(20.00,15.48){\includegraphics[scale=0.3]{yury.jpg}}
```

`\input{имя_файла}` — команда, вставляемая в файл `*.tex` и предназначенная для того, чтобы при компиляции файла вместо команды

`\input`{файл}

перейти к компиляции файла «`файл`» так, как если бы содержащийся в нём код был вставлен в исходный файл.

Например, если файл, допустим, `Proba.tex` начинается командой

```
\input texsample.tex
```

то сначала компилируется содержимое файла `texsample.tex`, после чего процесс продолжается компиляцией содержимого файла `Proba.tex`.

`\int` — команда печати обозначения интеграла $\int$.

J

K

L

`\label{...}` — команда определения метки. Вместо троеточия надо записать цепочку латинских букв (другие символы лучше не использовать). При подготовке презентаций с использованием рассматриваемой технологии следует использовать введенную нами команду `\mylabel{...}`, например, `\mylabel{abcd}`. При использовании последней команды определяется еще и мишень `HTabcd` для гиперссылок, осуществляемых командами `\hyperlink`, `\href` и др.

\LARGE, \Large, \large — примеры команд указания размера шрифта:

\Huge, \huge, \LARGE, \Large, \large, \normalsize, \small.

Например, {\Huge Самый большой}, {\huge поменьше}, {\LARGE ещё меньше}, {\Large ещё меньше}, {\large ещё меньше}, {\normalsize ещё меньше}, {\small ещё меньше}, {\tiny совсем мелко}.

`law` — имя окружения типа «теорема» — имя окружения для оформления законов, определенное в файле `texsample.tex`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании \newtheorem.

Автоматизировать создание окружений типа «теорема» можно с помощью команды \ThmEnvCom. Как для любого другого окружения, с окружением `law` связан одноименный счетчик, причем после выполнения `\begin{law}` значение счетчика `law` увеличивается на 1.

Например, компиляция фрагмента

```
Начальное значение счетчика^law равно^\arabic{law}.
    \begin{law}
Сила тока в участке цепи прямо пропорциональна напряжению
на концах этого проводника
и обратно пропорциональна его сопротивлению.
    \end{law}
Теперь значение счетчика law равно 0.
```

дает следующий результат:

Начальное значение счетчика `law` равно 0.

Закон 1. *Сила тока в участке цепи прямо пропорциональна напряжению на концах этого проводника и обратно пропорциональна его сопротивлению.*

Теперь значение счетчика `law` равно 1.

`\left` — часть команды для создания скобок, «растягивающихся вверх». Другим обязательным компонентом является `\right`.

`$\left[\frac{x}{y}\right)$` после компиляции даёт $\left[\frac{x}{y}\right)$;

`$\left.\frac{x}{y}\right)$` после компиляции даёт $\left.\frac{x}{y}\right)$;

`$\left[\frac{x}{y}\right.$` после компиляции даёт $\left[\frac{x}{y}\right.$.

`\lefteqn{`*формула*`}` — команда, обеспечивающая такую печать аргумента «*формула*», что LaTeX под неё не отводит места.

Например, компиляция кода

```
XYZ\lefteqn{abc}Ty $ST\lefteqn{abcde}ypC$
```

приводит к такому результату:

XYZ*abc*Ty ST*abcde*ypC

В частности, аргумент команды `\lefteqn` всегда компилируется в математической моде.

`lmm` — имя окружения для оформления определений, определенное в файле `texsample.tex`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании **\newtheorem**.

Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`. Как для любого другого окружения, с окружением `lmm` связан одноименный счетчик, причем после выполнения `\begin{lmm}` значение счетчика `lmm` увеличивается на 1.

`lmm` — имя окружения для оформления определений, определенное в файле `texsample.tex`.

Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#lmm# равно~\arabic{lmm}.
 \begin{lmm}
Если треугольник является равнобедренным, то
углы при его основании равны по величине.
 \end{lmm}
Теперь значение счетчика~\verb#lmm# равно~\arabic{lmm}.
```

дает следующий результат:

Начальное значение счетчика `lmm` равно 0.

Лемма 1. *Если треугольник является равнобедренным, то углы при его основании равны по величине.*

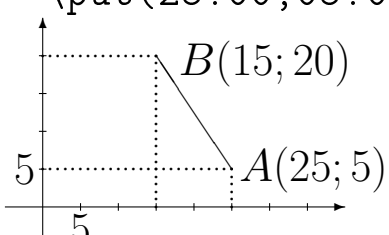
Теперь значение счетчика `lmm` равно 1.

`\line(a,b){c}` — команда для рисования отрезков в окружении `picture`. Здесь c — это проекция изображаемого отрезка на ось абсцисс, если отрезок расположен наклонно или ортогонально по отношению к оси ординат. Если же этот отрезок параллелен оси ординат, то c равен проекции изображаемого отрезка на *ось ординат*. Далее, a, b — целые числа, причем, во-первых, $\begin{cases} -6 \le a \le 6, \\ -6 \le b \le 6. \end{cases}$

Во-вторых, если a и b не равны 0, числа a и b должны быть взаимно простыми числами.

Например, команда

`\put(25.00,05.00){\line(-2,3){10}}` изобразит отрезок AB:



`\loop` — команда для организации цикла:

`\loop` *Безусловная часть цикла*
`\ifnum` *Условие*
    Условная часть цикла
`\repeat`

Обратите внимание на замену команды `\fi`, «замыкающей» действие команды `\ifnum`, на команду `\repeat`.

M

`magenta` — пурпурный цвет. Используется, например в составе команды
`{\color{magenta}пурпурный цвет}`.

`\makebox(w,u)[ab]{Ф}` — команда создания «бокса» размера $w \times u$, т.е. шириной w единиц длины и высотой u единиц длины. Переменная a может принимать одной из трех значений:

– l или r, тогда содержимое Ф «бокса» прижимается к левому или, соответственно, к правому краю этого «бокса»;

– c, тогда содержимое Ф центрируется в «боксе» по горизонтали.

Переменная b также может принимать одной из трех значений:

– t или b, тогда содержимое Ф прижимается в боксе к верхнему (top) или, соответственно, к нижнему (bottom) краю этого «бокса»;

– c, тогда содержимое Ф центрируется в «боксе» по вертикали.

`\fbox{\makebox(20,15)[lt]{a}}` 

, `\fbox{\makebox(20,15)[lc]{a}}` 

,

`\fbox{\makebox(20,15)[lb]{a}}` 

, `\fbox{\makebox(20,15)[rb]{a}}` 

,

`\fbox{\makebox(20,15)[ct]{a}}` 

, `\fbox{\makebox(20,15)[cc]{a}}` 

`\MakeNumbb{имя_команды}{`n`}` — команда, определенная в файле `MakeNumbb.inp` (загружается командой `\input MakeNumbb.inp`), которая создает команду `\имя_команды`, выдающую число n, причем n может задаваться и как содержимое **счетчика** или **регистра**, кроме `\the\CountSca` и `\the\CountTmp`.

Требуется, чтобы был определён **регистр** `\CountTmp`.

Этот регистр уже определён в случае использования преамбул `testsample.tex`, `StartTestBigScr.tex`.

Подробнее см. `00Pattern.pdf`, раздел **Команда для запоминания числа, в частности, текущего значения счётчика или регистра.**

`\MakeSimpleNumb{`n`}` — команда, определенная в файле `MakeSimpleNumb.inp` (загружается командой `\input MakeSimpleNumb`. которая при $1 \leqslant n \leqslant 168$ печатает n-е простое число, причем n может задаваться и как содержимое **счетчика** или **регистра**, кроме `\the\CountSca` и `\the\CountTmp`.

При других значениях n, не кратных 168, будут печататься простые числа с номерами, вычисляемыми по модулю 168, т.е. с номерами, равными остатку от деления на 168.

Для чисел, кратных 168, будет печататься 168-е простое число, т.е. число 997.

`\MakExecisePdf{условие}{Ответ}` — команда для **автоматизации оформления задач** для самостоятельного решения.

Внутри условия и решения не должно быть несколько абзацев, т.е. команды `\par` или пустой строки).

Если в ответе надо перейти к следующей странице так, чтобы условие вновь было напечатано, следует набрать команду `\MakeExNewpage`.

Если в ответе надо перейти к следующей странице так, чтобы условие не было напечатано, следует набрать команду `\ClearExNewpage`.

`\mbox{Ф}` — команда, которая «сшивает» цепочку букв (включая пробел) Ф в единый бокс. Например, если слово или формулу поместить в `\mbox`, то это слово или формула не могут быть разорваны при переносе на новую строку. В частности, если слово «`многословие`» окажется в конце строки, то при переносе на новую строку это слово может быть разорвано, например, на «`многосло-`» и «`вие`». Но если это слово использовать в виде

`\mbox{многословие}`,

то либо это слово целиком будет перенесено на новую строку, либо останется в предыдущей строке, даже если правый край этого слова «вылезет за пределы» строки.

`\medskip` — команда для увеличения расстояния между абзацем, в котором встретилась эта команда, и предыдущим абзацем.

Расстояние увеличивается на несколько ме́ньшую величину, чем при использовании команды `\bigskip`.

Здесь была вставлена команда `\medskip`.

Здесь была вставлена команда `\bigskip`.

Вставка заранее определённого расстояния между абзацами осуществляется командой `\vspace`. Например, перед этим абзацем была вставлена команда `\vspace{3ex}`.

\mp — знак $\mp$. Работает только в математической моде.
Аналогом является \pm — знак $\pm$.

`\multiply\`*идентификатор_регистра* `by` n — команда, в результате применения которой значение регистра с именем `\`*идентификатор_регистра* станет равным произведению начального значения регистра с именем `\`*идентификатор_регистра* на число n.

Специальное слово `by` является необязательным.

Например, выполнение кода

```
\newcount\Cadvance
Начальное значение регистра равно \Cadvance=5
\the\Cadvance,
потом \multiply\Cadvance 4  \the\Cadvance,
а теперь \multiply\Cadvance -2  $(\the\Cadvance)$
```

приведёт к следующему результату:

Начальное значение регистра равно 5, потом 15, а теперь (−30).

`\mylabel{...}` — доопределенная нами команда, которая вводит метку и одновременно, мишень для гиперссылок, см. `\label`.

При использовании команды `\mylabel{имя_метки}` автоматически определяется **гипермишень** с именем `HTимя_метки`.

Например, в результате компиляции команды `\mylabel{AboutMylabe` будет,
во-первых, определена метка `AboutMylabel`, на которую можно будет сослаться командами `\ref` и `\pageref`,
во-вторых, определена **гипермишень** `HTAboutMylabel`, на которую можно осуществить переход из любого места того же документа с помощью команды `\hyperlink` или из другого документа с помощью команды `\href`.

`\mylabel{...}` — доопределенная нами команда, которая вводит метку и одновременно, мишень для гиперссылок, см. `\label`.

При использовании команды `\mylabel{имя_метки}` автоматически определяется **гипермишень** с именем `HTимя_метки`.

Отметим, что метка (в отличие от гипермишени) обеспечивает ссылку не на точку, где находится команда `\label`, а в ближайшую предстоящую точку, в которой в последний раз с помощью команды `\refstepcounter` изменялось значение какого-либо счётчика.

Такие изменения счётчика автоматически происходят при использовании команд `\PdfSection`, `\PdfSubSection`, `\PdfSubSubSection`, `\ThmEnvCom` (точнее, при первом использовании соответствующей `\ComText...`), `\PrimCom` (точнее, при первом использовании соответствующей `\ComTextPrim...`) `\MakExcisePdf` и применении окружений `\begin{окружение}...\end{окружение}`, например, `\begin{equation}...\end{equation}`.

`\multicolumn{`n`}{`*знак*`}{элемент}` — команда для введения в строку таблицы, созданной **окружениями** `tabular` и `array`, элемента, занимающего n столбцов.

Здесь параметр *знак* может приимать значения:

`l` — выровнять элемент по левому краю поля из n столбцов;

`c` — выровнять элемент по центру поля из n столбцов;

`r` — выровнять элемент по правому краю поля из n столбцов.

Наличие символа `|` слева или справа в параметре *знак* означает, что этот элемент имеет соответствующую вертикальную границу.

```
\begin{tabular}{l|crcc}
1 & 2 & 3 & 4 & 5\\
11 & 12 & 13 & 14 & 15\\
311 & 322 & 333 & 344 & 355\\
\end{tabular}
```

1	2	3	4	5
11	12	13	14	15
311	322	333	344	355

Здесь команда `\multicolumn` пока не использована.

`\multicolumn{`n`}{`*знак*`}{элемент}` — команда для введения в строку таблицы, созданной **окружениями** `tabular` и `array`, элемента, занимающего n столбцов.

Здесь параметр *знак* может приимать значения:

`l` — выровнять элемент по левому краю поля из n столбцов;

`c` — выровнять элемент по центру поля из n столбцов;

`r` — выровнять элемент по правому краю поля из n столбцов.

Наличие символа `|` слева или справа в параметре *знак* означает, что этот элемент имеет соответствующую вертикальную границу.

```
\begin{tabular}{l|crcc}
1 & 2 & 3 & 4 & 5\\
11&12& \multicolumn{2}{c}{44} &15\\
311 & 322 & 333 & 344 & 355\\
\end{tabular}
```

1	2	3	4	5
11	12	44		15
311	322	333	344	355

`\multicolumn{`n`}{`*знак*`}{элемент}` — команда для введения в строку таблицы, созданной **окружениями** `tabular` и `array`, элемента, занимающего n столбцов.

Здесь параметр *знак* может приимать значения:

`l` — выровнять элемент по левому краю поля из n столбцов;

`c` — выровнять элемент по центру поля из n столбцов;

`r` — выровнять элемент по правому краю поля из n столбцов.

Наличие символа `|` слева или справа в параметре *знак* означает, что этот элемент имеет соответствующую вертикальную границу.

```
\begin{tabular}{l|crcc}
1 & 2 & 3 & 4 & 5\\
11&12& \multicolumn{2}{c|}{44}&15\\
311 & 322 & 333 & 344 & 355\\
\end{tabular}
```

1	2	3	4	5
11	12	44		15
311	322	333	344	355

`\multicolumn{`n`}{`*знак*`}{элемент}` — команда для введения в строку таблицы, созданной **окружениями** `tabular` и `array`, элемента, занимающего n столбцов.

Здесь параметр *знак* может приимать значения:

`l` — выровнять элемент по левому краю поля из n столбцов;

`c` — выровнять элемент по центру поля из n столбцов;

`r` — выровнять элемент по правому краю поля из n столбцов.

Наличие символа `|` слева или справа в параметре *знак* означает, что этот элемент имеет соответствующую вертикальную границу.

```
\begin{tabular}{l|crcc}
1 & 2 & 3 & 4 & 5\\
11&12& \multicolumn{2}{l|}{44}&15\\
311 & 322 & 333 & 344 & 355\\
\end{tabular}
```

1	2	3	4	5
11	12	44		15
311	322	333	344	355

`\multicolumn{`n`}{`*знак*`}{элемент}` — команда для введения в строку таблицы, созданной **окружениями** `tabular` и `array`, элемента, занимающего n столбцов.

Здесь параметр *знак* может приимать значения:

`l` — выровнять элемент по левому краю поля из n столбцов;

`c` — выровнять элемент по центру поля из n столбцов;

`r` — выровнять элемент по правому краю поля из n столбцов.

Наличие символа `|` слева или справа в параметре *знак* означает, что этот элемент имеет соответствующую вертикальную границу.

```
\begin{tabular}{l|crcc}
1 & 2 & 3 & 4 & 5\\
11&12& \multicolumn{2}{r|}{44}&15\\
311 & 322 & 333 & 344 & 355\\
\end{tabular}
```

1	2	3	4	5
11	12		44	15
311	322	333	344	355

`\multicolumn{`n`}{`*знак*`}{элемент}` — команда для введения в строку таблицы, созданной **окружениями** `tabular` и `array`, элемента, занимающего n столбцов.

Здесь параметр *знак* может приимать значения:

`l` — выровнять элемент по левому краю поля из n столбцов;

`c` — выровнять элемент по центру поля из n столбцов;

`r` — выровнять элемент по правому краю поля из n столбцов.

Наличие символа `|` слева или справа в параметре *знак* означает, что этот элемент имеет соответствующую вертикальную границу.

```
\begin{tabular}{l|crcc}
1 & 2 & 3 & 4 & 5\\
11&12&\multicolumn{2}{|r|}{44}&15\\
311 & 322 & 333 & 344 & 355\\
\end{tabular}
```

1	2	3	4	5
11	12		44	15
311	322	333	344	355

`\multicolumn{`n`}{`*знак*`}{элемент}` — команда для введения в строку таблицы, созданной **окружениями** `tabular` и `array`, элемента, занимающего n столбцов.

Здесь параметр *знак* может приимать значения:

`l` — выровнять элемент по левому краю поля из n столбцов;

`c` — выровнять элемент по центру поля из n столбцов;

`r` — выровнять элемент по правому краю поля из n столбцов.

Наличие символа `|` слева или справа в параметре *знак* означает, что этот элемент имеет соответствующую вертикальную границу.

```
\begin{tabular}{l|crcc}
1 & 2 & 3 & 4 & 5\\
11&12&\multicolumn{2}{|r||}{44}&15\\
311 & 322 & 333 & 344 & 355\\
\end{tabular}
```

1	2	3	4	5
11	12		44	15
311	322	333	344	355

`\multicolumn{`n`}{`*знак*`}{элемент}` — команда для введения в строку таблицы, созданной **окружениями** `tabular` и `array`, элемента, занимающего n столбцов.

Здесь параметр *знак* может приимать значения:

`l` — выровнять элемент по левому краю поля из n столбцов;

`c` — выровнять элемент по центру поля из n столбцов;

`r` — выровнять элемент по правому краю поля из n столбцов.

Наличие символа `|` слева или справа в параметре *знак* означает, что этот элемент имеет соответствующую вертикальную границу.

```
\begin{tabular}{l|crcc}
1 & 2 & 3 & 4 & 5\\
11 & \multicolumn{3}{|r||}{44}&15\\
311 & 322 & 333 & 344 & 355\\
\end{tabular}
```

1	2	3	4	5
11			44	15
311	322	333	344	355

`\multicolumn{`n`}{`*знак*`}{элемент}` — команда для введения в строку таблицы, созданной **окружениями** `tabular` и `array`, элемента, занимающего n столбцов.

Здесь параметр *знак* может приимать значения:

`l` — выровнять элемент по левому краю поля из n столбцов;

`c` — выровнять элемент по центру поля из n столбцов;

`r` — выровнять элемент по правому краю поля из n столбцов.

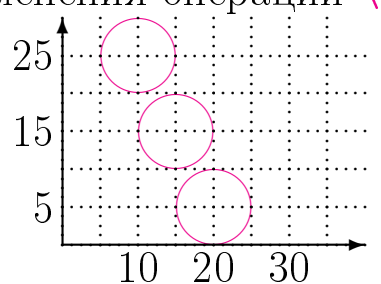
Наличие символа `|` слева или справа в параметре *знак* означает, что этот элемент имеет соответствующую вертикальную границу.

```
\begin{tabular}{l|crcc}
1 & 2 & 3 & 4 & 5\\
11 & \multicolumn{3}{r||}{44}&15\\
311 & 322 & 333 & 344 & 355\\
\end{tabular}
```

1	2	3	4	5
11			44	15
311	322	333	344	355

`\multiput(x,y)(dx,dy){n}{Ф}` — команда многократного повторения фигуры Ф. Первоначально фигура помещается в точку с координатами (x, y), последующие n копий фигуры Ф смещаются каждый раз на dx единиц длины по горизонтали (положительное направление — вправо) и на dy единиц длины по вертикали (положительное направление — вверх).

На рисунке пурпурным (magenta) цветом изображен результат применения операции `\multiput(10,25)(5,-10){3}{\circle{10}}`



N

`\newcommand` — команда для создания новых команд.

Эта команда применяется только в случае, если определяется новая команда.

Если же необходимо переопределить уже существующую команду, следует использовать команду `\renewcommand` или `\def`.

Пробелы, следующие непосредственно за командой, компилятор TEXвоспринимает как знак окончания имени команды, т.е. эти пробелы на печати игнорируются. Например,

```
\newcommand{\Ab}{пере}
\Ab ехал, \Ab шел, \Ab двинул...
```

после компиляции даёт:

переехал, перешел, передвинул...

а не

пере ехал, пере шел, пере двинул...

Рассмотрим формат команды `\newcommand`.

`\newcommand` — команда для создания новых команд.

Формат команды:

```
\newcommand{\имя_команды}{текст_команды}
```

или

```
\newcommand{\имя_команды}[число_параметров]{текст_команды}
```

Например, следующим кодом определяется команда `\ComA`:

```
\newcommand{\ComA} {такая, что }
```

Тогда результатом компиляции кода:

```
Если вещь \ComA она хорошо заметна, то ее надо демонстрировать
а если мысль \ComA она впечатляет, то ее надо пропагандироват
```

будет текст

`\newcommand` — команда для создания новых команд.

Формат команды:

```
\newcommand{\имя_команды}{текст_команды}
```

или

```
\newcommand{\имя_команды}[число_параметров]{текст_команды}
```

Например, следующим кодом определяется команда `\ComA`:

```
\newcommand{\ComA} {такая, что }
```

Тогда результатом компиляции кода:

```
Если вещь \ComA она хорошо заметна, то ее надо демонстрировать
а если мысль \ComA она впечатляет, то ее надо пропагандировать
```

будет текст

«Если вещь такая, что она хорошо заметна, то ее надо демонстрировать, а если мысль такая, что она впечатляет, то ее надо пропагандировать».

\newcommand — команда для создания новых команд.

Формат команды:

```
\newcommand{\имя_команды}{текст_команды}
```

или

```
\newcommand{\имя_команды}[число_параметров]{текст_команды}
```

Следующим кодом определяется команда \ComD:

```
\newcommand{\ComD}[3] {если #1 такая, что она #2, то ее #3}
```

Тогда результатом компиляции кода:

```
E\ComD{вещь}{хорошо заметна}{надо демонстировать}, а
e\ComD{мысль}{впечатляет}{пропагандировать}
```

будет текст

`\newcommand` — команда для создания новых команд.

Формат команды:

```
\newcommand{\имя_команды}{текст_команды}
```

или

```
\newcommand{\имя_команды}[число_параметров]{текст_команды}
```

Следующим кодом определяется команда `\ComD`:

```
\newcommand{\ComD}[3] {если #1 такая, что она #2, то ее #3}
```

Тогда результатом компиляции кода:

```
E\ComD{вещь}{хорошо заметна}{надо демонстировать}, а
e\ComD{мысль}{впечатляет}{пропагандировать}
```

будет текст

«Если вещь такая, что она хорошо заметна, то ее надо демонстировать, а если мысль такая, что она впечатляет, то ее пропагандировать»

`\newcount\`*идентификатор_регистра* — команда для определения нового регистра с именем `\`*идентификатор_регистра*.

Например, команда
`\newcount\Aaa` определяет регистр `\Aaa`.

К регистра (в отличие от счётчика LaTeX, определяемого командой `\newcounter` и в идентификаторе которого отсутствует `\`) можно применять операции `\advance`, `\multiply`, `\divide`. При этом печать значения регистра осуществляется не командой `\arabic` (тем более не `\roman` и `\Roman`), а командой `\the`*идентификатор_регистра*. Например, команда `\the\Pqrt` обеспечит печать значения регистра `\Pqrt`.

`\newcounter` — команда для создания нового счётчика.
Синтаксис команды:

```
\newcounter{имя_счётчика}
```

`\newlength{команда-длина}` — команда для определния команды-длины, т.е. команды, обозначающей длину.

Например, `\newlength{\MyLength}` определяет новую длину `\MyLength`.

`\newline` — команда принудительного перехода к новой строке. При этом следующая строка начинается без абзацного отступа и без вставки дополнительных вертикальных отступов от предыдущей строки.

Например, после компиляции текста `Первая строка.\newline Втора`
`Новый абзац.`
получим

Первая строка.
Вторая строка.

Новый абзац.

`\newtheorem{...}{...}` — команда для введения нового окружения типа «теорема». В преамбуле из файла `texsample.tex` определены окружения:

```
\newtheorem{lmm}{\noindent \bf Лемма}%[section]
\newtheorem{thm}{\noindent \bf Теорема}%[section]
\newtheorem{prim}{\noindent \bf Пример}%[section]
\newtheorem{zad}{\noindent \bf Задача}%[section]
\newtheorem{quest}{\noindent \bf Вопрос}%[section]
\newtheorem{cons}{\noindent \bf Следствие}%[section]
\newtheorem{zam}{\noindent \bf Замечание}%[section]
\newtheorem{sogl}{\noindent \bf Соглашение}%[section]
\newtheorem{aim}{\noindent \bf Цель}%[section]
\newtheorem{law}{\noindent \bf Закон}%[section]
\newtheorem{regul}{\noindent \bf Правило}%[section]
```

`\newpage` — команда принудительного перехода к новой странице.

`\noindent` — команда подавления абзацного отступа.

`\No` знак номера № (лучше в математической моде).

`\normalsize` — команда переключения размера шрифта.

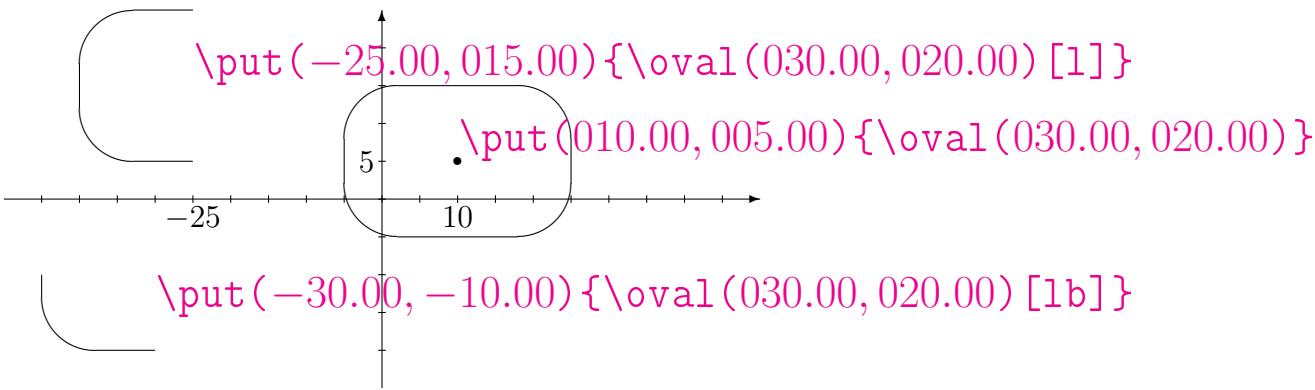
Например, {\Huge Самый большой}, {\huge поменьше}, {\LARGE ещё меньше}, {\Large ещё меньше}, {\large ещё меньше}, {\normalsize ещё меньше}, {\small ещё меньше}, {\tiny совсем мелко}.

O

`\or` — фрагмент команды `\ifcase` выбора из нескольких вариантов.

`\oval(a,b)` — команда рисования прямоугольника со скруглёнными углами, где a — ширина овала, b — высота овала.

Результат применения команд



Возможно применение необязательных параметров p, q: `\oval(`a, b`)[`pq`]`,
где $p \in \{l, c, r\}$, $q \in \{b, c, t\}$.

`\overbrace{`*формула1*`}^{`*формула2*`}` — команда, действующая в математической моде, ставящая ***над*** «*формула1*» фигурную скобку, завершаюся сверху выражением «*формула2*».

Часть этой команды вида `^{`*формула_2*`}` является необязательной.

Например, код

`$x+\overbrace{y+z}+t$` и `$x+\overbrace{y+z}^{\alpha}+t$`

в результате компиляции даёт такой результат:

$x+\overbrace{y+z}+t$ и $x+\overbrace{y+z}^{\alpha}+t$

P

`\pageref{имя_метки}` — команда ссылки на страницу, на которой находится команда `\label{имя_метки}` или `\mylabel{имя_метки}`.

Метка ставится командами `\label` и `\mylabel`.

В отличие от команды `\ref`, команда `\pageref` печатает непосредственно номер той страницы, на которой находится `\label{имя_метки}` или `\mylabel{имя_метки}`, и никак не связана с изменениями значений других счётчиков.

`\par` — команда для перехода к новому абзацу. Равносильна вставке пустой строки.

В результате компиляции кода LaTeX:

«`Первый абзац. Он может занимать несколько строк. После него поставили texttt{\symbol{92}par}.\par`
`Второй абзац. Он тоже может занимать несколько строк. После него оставили пустую строку.`

`Третий абзац.»`
получим следующий результат:

Первый абзац. Он может занимать несколько строк. После него поставили `\par`.

Второй абзац. Он тоже может занимать несколько строк. После него оставили пустую строку.

Третий абзац.

`\parbox{w}{Ф}` — команда создания бокса шириной не более w, с явным указанием единицы длины: `pt`, `mm`, `cm` и др. При этом цепочка слов Ф форматируется в абзац шириной w единиц длины. Например, команда

```
\parbox{30mm}{Я сформировал абзац шириной не более 30 mm}
```

после компиляции приведет к результату Я сформировал абзац шириной не более 30 mm, в данном случае выглядящим довольно странно...

`\partial` — команда для печати символа ∂.

`\PdfSection{название_раздела}{0 или 1}{}{}{}{}{}{}` — команда секционирования, адаптированная к задаче создания слайдов.

Аналогично определяются команды

`\PdfSubSection` (ей соответствует счётчик `subsection`),

`\PdfSubSubSection` (ей соответствует счётчик `subsubsection`).

При компиляции команды

```
\PdfSection{Слоны и мыши}{0}{}{}{}{}{}{}
```

во-первых, принудительно закрывается текущая страница (т.е. автоматически выполняется `\newpage`),

во-вторых, счётчик `section` увеличивается на 1,

в-третьих, на новой странице печатается номер раздела и его название (в данном примере название «Слоны и мыши»),

в-четвёртых, в файл с оглавлением (т.е. `*.toc`) записывается новый номер раздела, его название и номер страницы, с которой он начинается.

`\PdfSection{название_раздела}{0 или 1}{}{}{}{}{}{}` — команда секционирования, адаптированная к задаче создания слайдов.

Аналогично определяются команды
`\PdfSubSection` (ей соответствует счётчик `subsection`),
`\PdfSubSubSection` (ей соответствует счётчик `subsubsection`).

Если второй аргумент равен 1, а не 0, т.е. при компиляции команды
`\PdfSection{Слоны и мыши}{1}{}{}{}{}{}{}{}`
во-первых, принудительно закрывается текущая страница (т.е. автоматически выполняется `\newpage`), но счётчик `section` ***больше не изменяется(!)***, во-вторых, печатается новый номер раздела и его название (в данном примере название «Слоны и мыши»).

В файл с оглавлением больше ничего не заносится.

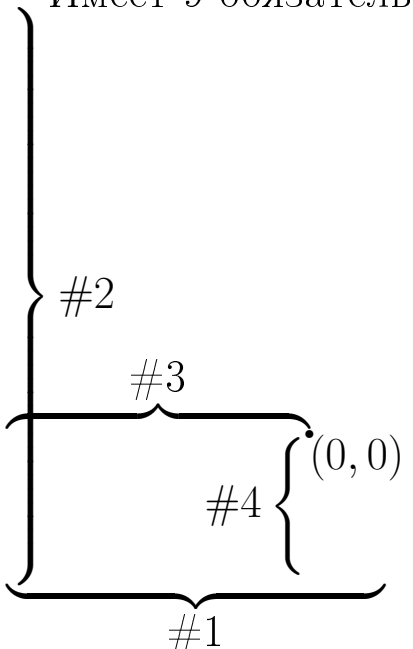
`\phantom{w}` — команда, которая печатает слово «w» «невидимым образом».

`<<Например, этот текст>>` напечатается как «Например, этот текст».

`<<Например, \phantom{этот} текст>>` напечатается как «Например, текст».

`\PictText` — команда вывода текста справа от рисунка, определённая в `texsample.tex`

Имеет 9 обязательных параметров:



#1 — натуральное число, означающее ширину рисунка в миллиметрах;

#2 — натуральное число, означающее высоту рисунка в миллиметрах;

#3 — натуральное число, означающее смещение вправо точки с координатами $(0, 0)$ относительно левого нижнего края рисунка;

#4 — натуральное число, означающее смещение вверх точки с координатами $(0, 0)$ относительно левого нижнего края рисунка;

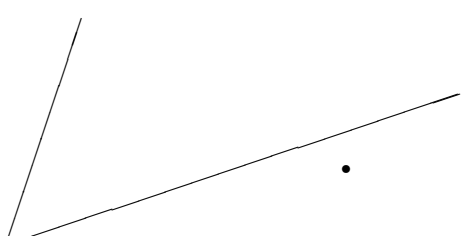
#5 — система команд для окружения **picture**: **\put**... и др.;

#6 — текст, печатаемый справа от рисунка, #7,#8,#9 — свободны.

Для вывода текста слева от рисунка применяется **TextPict**.

`\PictText` — команда вывода текста справа от рисунка.

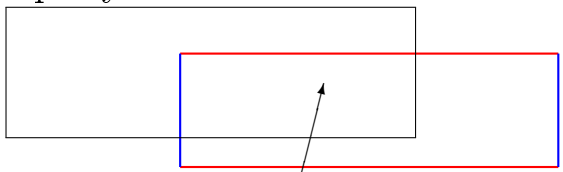
```
\PictText{60}{30}{45}{10}{
 \put(-45.00,-10.00){\line( 3, 1){060.00}}
 \put(-45.00,-10.00){\line( 1, 3){010.00}}
 \put(000.00,000.00){\circle*{1}}
}{Текст может содержать несколько абзацев.\par
  Использованы команды
  \symbol{92}put, \symbol{92}line, \symbol{92}circle,
  \symbol{92}symbol.
}{}{}{}
```



Текст может содержать несколько абзацев.
Использованы команды
\put, **\line**, **\circle**, **\symbol**.

`picture` — окружение для построения рисунка.

```
\begin{picture}(aw,bh)(cw,dh)
  команды рисования
\end{picture}
```



Здесь `aw` и `bh` — ширина и длина «бокса» (прямоугольника), выделяемого компилятором TEXa на странице для рисунка. Единицей длины является значение переменной `\unitlength`. Необязательные параметры `cw` и `dh` означают сдвиг рисунка на $-$`cw` единиц длины по горизонтали и на $-$`dh` единиц длины по вертикали.

```
\fbox{\begin{picture}(50.00,15.00)(-20.00,05.00)
       \put(00.00,00.00){{\color{red}\line( 1, 0){50.00}}}
       \put(00.00,15.00){{\color{red}\line( 1, 0){50.00}}}
       \put(00.00,00.00){{\color{blue}\line( 0, 1){15.00}}}
       \put(50.00,00.00){{\color{blue}\line( 0, 1){15.00}}}
      \end{picture}}
```

`pmatrix` — окружение для матрицы в круглых скобках.

В результате компиляции кода

```
$\begin{pmatrix}
1 & 2\\ 3 & 4\\
\end{pmatrix},\quad
\begin{pmatrix}
1 & 2\\ 3 & 4\\ 5 & 6\\
\end{pmatrix},\quad
\begin{pmatrix}
1 & 2 & 3\\ 4 & 5 & 6\\
\end{pmatrix}.$
```

получим

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$

`prim` — имя окружения для оформления примеров, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`, а для примеров — `\PrimCom`. Как для любого другого окружения, с окружением `prim` связан одноименный счетчик, причем после выполнения `\begin{prim}` значение счетчика `prim` увеличивается на 1. Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#prim# равно~\arabic{prim}.
 \begin{prim}
Вычислите $\arcsin{0,8}+\arcsin{0,6}$.
 \end{prim}
Теперь значение счетчика~\verb#prim# равно~\arabic{prim}.
```

дает следующий результат:

Начальное значение счетчика `prim` равно 0.

Пример 1. *Вычислите* $\arcsin 0,8 + \arcsin 0,6$.

Теперь значение счетчика `prim` равно 1.

`\PrimCom` — команда для **автоматизации оформления примеров.**

\pm — знак $\pm$. Работает только в математической моде.
Аналогом является \mp — знак $\mp$.

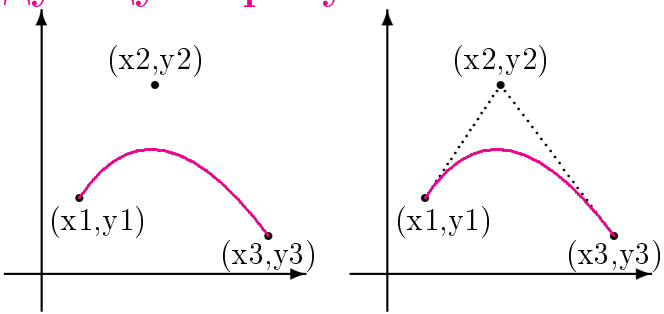
`\put(x,y){Ф}` — команда для помещения фигуры Ф в точку с координатами x, y. Начало координат находится в левом нижнем углу бокса, отведенного под рисунок. Например, команда

```
\put(20.00,15.00){\circle{40}}
```

изобразит окружность диаметром 40 (единиц длины) с центром в точке с координатами $(20, 15)$.

Q

`\qbezier(x1,y1)(x2,y2)(x3,y3)` — команда из пакета, подгружаемого стилевым файлом `curves.sty`. Эта команда изображает **следующую кривую**:



\quad — вставка «большого пробела»:
(<<большого\quad пробела>>).

\qquad — вставка «очень большого   пробела»:
(<<очень большого\qquad пробела>>).

`quest` — имя окружения для оформления вопросов, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании `\newtheorem`.

Как для любого другого окружения, с окружением `quest` связан одноименный счетчик, причем после выполнения `\begin{quest}` значение счетчика `quest` увеличивается на 1.

`quest` — имя окружения для оформления вопросов, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`.

Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#quest# равно~\arabic{quest}.
 \begin{quest}
Как связаны длины сторон треугольника с величинами его углов?
 \end{quest}
Теперь значение счетчика~\verb#quest# равно~\arabic{quest}.
```

дает следующий результат:

Начальное значение счетчика `quest` равно 0.

Вопрос 1. *Как связаны длины сторон треугольника с величинами его углов?*

Теперь значение счетчика `quest` равно 1.

R

`regul` — имя окружения типа «теорема» — имя окружения для оформления правил, определенное в файле `texsample.tex`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании \newtheorem.

Автоматизировать создание окружений типа «теорема» можно с помощью команды \ThmEnvCom. Как для любого другого окружения, с окружением `regul` связан одноименный счетчик, причем после выполнения `\begin{regul}` значение счетчика `regul` увеличивается на 1.

Например, компиляция фрагмента

```
Начальное значение счетчика^regul равно^\arabic{regul}.
    \begin{regul}
Если $f$ и^$g$ -- функции, бесконечно малые в точке^$a$ и
дифференцируемые в этой точке. Тогда
$\lim\limits_{x\to a}f(x)/g(x)= \lim\limits_{x\to a}f'(x)/g'(x)$.
    \end{regul}
Теперь значение счетчика regul равно 0.
```

дает следующий результат:

Начальное значение счетчика `regul` равно 0.

Правило 1. *Если f и g — функции, бесконечно малые в точке a и дифференцируемые в этой точке. Тогда*
$\lim\limits_{x\to a} f(x)/g(x) = \lim\limits_{x\to a} f'(x)/g'(x)$.

Теперь значение счетчика `regul` равно 1.

`\raisebox{`h`}{`содержимое_бокса`}` — команда, которая смещает «содержимое_бокса» по вертикали на h единиц (для h указание единиц измерения обязательно). Если $h > 0$, то сдвиг осуществляется вверх относительно базовой линии строки, а если $h < 0$ — вниз.

Например, этот текст есть результат компиляции кода:

`\raisebox{`h`}{содержимое_бокса}` — команда, которая смещает «содержимое_бокса» по вертикали на h единиц (для h указание единиц измерения обязательно). Если $h > 0$, то сдвиг осуществляется вверх относительно базовой линии строки, а если $h < 0$ — вниз.

Например, этот текст есть результат компиляции кода:

```
Например,\raisebox{3mm}{этот текст} \raisebox{-1.5ex}{есть}
\raisebox{-1ex}{результат} \raisebox{-0.5ex}{компиляции}
\raisebox{-0.5ex}{кода:}
```

`red` — красный цвет. Используется, например в составе команды `{\color{red}красный цвет}` (здесь это не гиперссылка).

`\ref{имя_метки}` — команда ссылки на значение метки с данным именем.

Метка ставится командами `\label` и `\mylabel`.

Значение метки равно значению счётчика, который был последним изменён с помощью `\refstepcounter` перед использованием команд `\label` и `\mylabel`.

Напечатать номер страницы, на которой находится команда `\label{имя_метки}` или `\mylabel{имя_метки}` можно с помощью команды `\pageref{имя_метки}`.

`\refstepcounter` — команда увеличения счетчика на 1. Например, `\refstepcounter{thm}` — увеличение на 1 счетчика `thm`. В отличие от `\addtocounter{thm}{1}` при использовании команды `\refstepcounter` использование последующей команды `\label` позволяет сделать корректную ссылку.

`\renewcommand` — команда для переопределения команд.

Эта команда применяется только в случае, если необходимо переопределить уже существующую команду.

Если же определяется новая команда, следует использовать команду `\newcommand` или `\def`.

Пробелы, следующие непосредственно за командой, компилятор TEXвоспринимает как знак окончания имени команды, т.е. эти пробелы на печати игнорируются. Например,

```
\newcommand{\Ab}{пере}
\Ab ехал, \Ab шел, \Ab двинул...
```

после компиляции даёт:

переехал, перешел, передвинул...

а не

пере ехал, пере шел, пере двинул...

Рассмотрим формат команды `\renewcommand`.

`\renewcommand` — команда для переопределения команд.

Формат команды:

```
\renewcommand{\имя_команды}{текст_команды}
```

или

```
\renewcommand{\имя_команды}[число_параметров]{текст_команды}
```

Пусть команда `\ComA` уже определена, но надо её переопределить. Это можно сделать командой:

```
\renewcommand{\ComA} {такая, что }
```

Тогда результатом компиляции кода:

```
Если вещь \ComA она хорошо заметна, то ее надо демонстрировать
а если мысль \ComA она впечатляет, то ее надо пропагандироват
```

будет текст

`\renewcommand` — команда для переопределения команд.

Формат команды:

```
\renewcommand{\имя_команды}{текст_команды}
```

или

```
\renewcommand{\имя_команды}[число_параметров]{текст_команды}
```

Пусть команда `\ComA` уже определена, но надо её переопределить. Это можно сделать командой:

```
\renewcommand{\ComA} {такая, что }
```

Тогда результатом компиляции кода:

```
Если вещь \ComA она хорошо заметна, то ее надо демонстрировать
а если мысль \ComA она впечатляет, то ее надо пропагандировать
```

будет текст

«Если вещь такая, что она хорошо заметна, то ее надо демонстрировать, а если мысль такая, что она впечатляет, то ее надо пропагандировать».

`\renewcommand` — команда для переопределения команд.

Формат команды:

```
\renewcommand{\имя_команды}{текст_команды}
```

или

```
\renewcommand{\имя_команды}[число_параметров]{текст_команды}
```

Допустим, была определена команда `\ComD`. Тогда её можно переопределить следующим кодом:

```
\renewcommand{\ComD}[3] {если #1 такая, что она #2, то ее #3}
```

Тогда результатом компиляции кода:

```
E\ComD{вещь}{хорошо заметна}{надо демонстировать}, а
e\ComD{мысль}{впечатляет}{пропагандировать}
```

будет текст

`\renewcommand` — команда для переопределения команд.

Формат команды:

```
\renewcommand{\имя_команды}{текст_команды}
```

или

```
\renewcommand{\имя_команды}[число_параметров]{текст_команды}
```

Допустим, была определена команда `\ComD`. Тогда её можно переопределить следующим кодом:

```
\renewcommand{\ComD}[3] {если #1 такая, что она #2, то ее #3}
```

Тогда результатом компиляции кода:

```
E\ComD{вещь}{хорошо заметна}{надо демонстировать}, а
e\ComD{мысль}{впечатляет}{пропагандировать}
```

будет текст

«Если вещь такая, что она хорошо заметна, то ее надо демонстировать, а если мысль такая, что она впечатляет, то ее пропагандировать»

`\repeat` — команда, завершающая действие команды организации цикла `\loop`.

`\right` — часть команды для создания скобок, «растягивающихся вверх». Другим обязательным компонентом является `\left`.

`$\left[\frac{x}{y}\right)$` после компиляции даёт $\left[\frac{x}{y}\right)$;

`$\left.\frac{x}{y}\right)$` после компиляции даёт $\left.\frac{x}{y}\right)$;

`$\left[\frac{x}{y}\right.$` после компиляции даёт $\left[\frac{x}{y}\right.$.

`\Roman{имя_счётчика}` — команда для печати значения счётчика заглавными римскими буквами.

В результате компиляции кода:

```
Значение сч\symbol{188}тчика section
равно \Roman{section}=
\roman{section}= \arabic{section}=
\Alph{section}= \alph{section}
```

получим

«Значение счётчика section равно II= ii= 2= B= b».

`\roman{имя_счётчика}` — команда для печати значения счётчика строчными римскими буквами.

В результате компиляции кода:

```
Значение сч\symbol{188}тчика section
        равно \roman{section}=
\Roman{section}= \arabic{section}=
  \Alph{section}= \alph{section}
```

получим

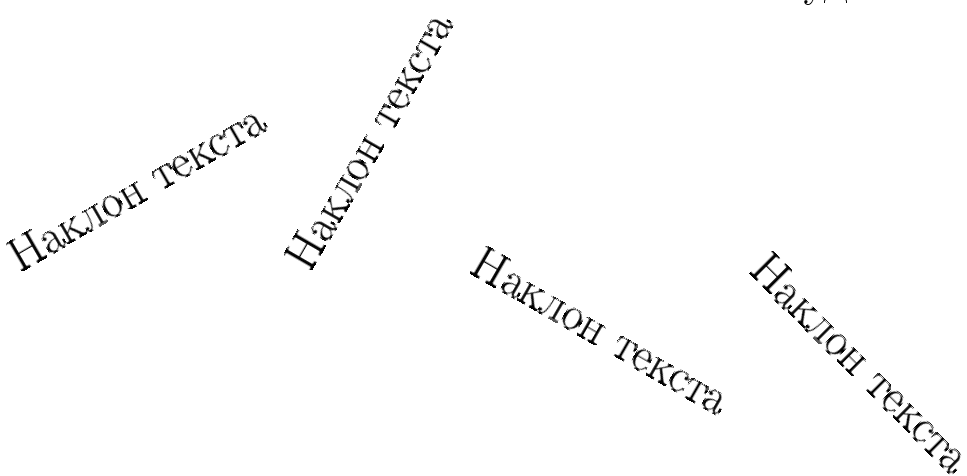
«Значение счётчика section равно ii= II= 2= B= b».

`\rotatebox{grad}{Текст}` — команда для печати строки «`Текст`», повёрнутого против часовой стрелки на угол `grad`. (стилевой файл `rotating`).

Например, результатом компиляции кода

```
\rotatebox{30}{Наклон текста} \rotatebox{60}{Наклон текста}
\rotatebox{-30}{Наклон текста}
\rotatebox{-45}{Наклон текста}
```

будет текст



`\rule[d]{w}{h}` — команда для вставки прямоугольника шириной w, высотой h, причем единицы измерения должны быть указаны явно: `mm`, `cm`, `pt`, `em`, `ex` и т.д. Если необязательный параметр d не указан или равен 0, то нижный край прямоугольника совпадает с нижним краем строки, где нижний край строки совпадает с нижним краем букв «а», «б» и др., при этом считается, что нижний край таких букв как «р», «у», «щ» и др. опускается ниже нижнего края строки. Если же в команде явно указано значение необязательного параметра d, то это нижний край прямоугольника «поднимается» над нижним краем строки на d, (при $d < 0$ — опускается ниже нижнего края строки). Например, команды

`\rule{3mm}{5mm}`, `\rule[2mm]{3mm}{5mm}`, `\rule[-2mm]{3mm}{5mm}`

после компиляции дают такой результат:  , ,  .

Если w или h равны 0, получаем «невидимую линейку».

`12,\quad 1\rule{4mm}{0mm}2` дает 12, 1 2

S

\S — знак **параграфа** §.

`\section` — одна из команд секционирования (выделения разделов).

`\section`, `\subsection`, `\subsubsection` — секция и подсекции, номер которых определяется счетчиком `section` и, соответственно, `subsection`, `subsubsection`.

Стандартный способ печати номеров секций и подсекций определяется командами `\thesection`, `\thesubsection`, `\thesubsubsection`. Допустим, значение счетчика `section` равно 3, а значение счетчика `subsection` равно 5. Тогда команда `\thesection` дает на печати III, а команда `\thesubsection` дает на печати III.5.

Печать номера любого счетчика, например, `section`, `subsection`, осуществляется командами

`\arabic{section}`, `\arabic{subsection}` — печать значения счетчиков `section`, `subsection` арабскими цифрами, например, `section`=3, `subsection`=5;

`\roman{section}`, `\roman{subsection}` — печать значения счетчика `section` римскими цифрами, например, `section`=iii, `subsection`=v.

При использовании `texsample.tex` доступны макрокоманды `\PdfSection`, `\PdfSubSection`, `\PdfSubSection`.

`\setcounter` — команда для присвоения счётчику некоторого значения.

Синтаксис команды:

```
\setcounter{имя_счётчика} {значение_счётчика}
```

`\setlength{команда-длина}{значение длины}` — команда присвоения команде-длине значения длины.

Например, если была определена новая длина командой `\newlength{\MyLength}`, то `\setlength{\MyLength}{12mm}` сделает синонимами `\MyLength` и `12 mm`.

Компилирование кода:

```
\newlength{\MyLength}
\setlength{\MyLength}{12mm}
\rule{\MyLength}{\MyLength}\quad \rule{12mm}{12mm}
```

приведёт к следующему результату:

`\settodepth{команда-длина}{Текстовая строка}` — команда для присвоения команде-длине (т.е. команде, обозначающей длину) значения, равного глубине текстовой строки, т.е. расстояния от базисной линии до нижнего края этой текстовой строки.

Здесь под командой-длиной понимается команда, обозначающая длину. Она должна быть введена заранее, например, командой `\setlength`.

`\settodepth{команда-длина}{Текстовая строка}` — команда для присвоения команде-длине (т.е. команде, обозначающей длину) значения, равного глубине текстовой строки, т.е. расстояния от базисной линии до нижнего края этой текстовой строки.

Здесь под командой-длиной понимается команда, обозначающая длину. Она должна быть введена заранее, например, командой `\setlength`.

`\settoheight{команда-длина}{Текстовая строка}` — команда для присвоения команде-длине значения, равного высоте текстовой строки, т.е. расстояния от базисной линии до верхнего края этой текстовой строки.

\settodepth{команда-длина}{Текстовая строка} — команда для присвоения команде-длине (т.е. команде, обозначающей длину) значения, равного глубине текстовой строки, т.е. расстояния от базисной линии до нижнего края этой текстовой строки.

Здесь под командой-длиной понимается команда, обозначающая длину. Она должна быть введена заранее, например, командой \setlength.

\settoheight{команда-длина}{Текстовая строка} — команда для присвоения команде-длине значения, равного высоте текстовой строки, т.е. расстояния от базисной линии до верхнего края этой текстовой строки.

\settowidth{команда-длина}{Текстовая строка} — команда для присвоения команде-длине (т.е. команде, обозначающей длину) значения, равного ширине текстовой строки.

`\setka` — команда для изображения координатной сетки в окружении `picture`.

Например, в результате компиляции фрагмента

```
{\unitlength=1mm
\begin{picture}(60,40)(-10,-20)
  \setka
  \put(-10.00,-20.00){\line ( 3, 2){060.00}}
  \put(050.00,-20.00){\line (-3, 2){060.00}}
\end{picture}}
```

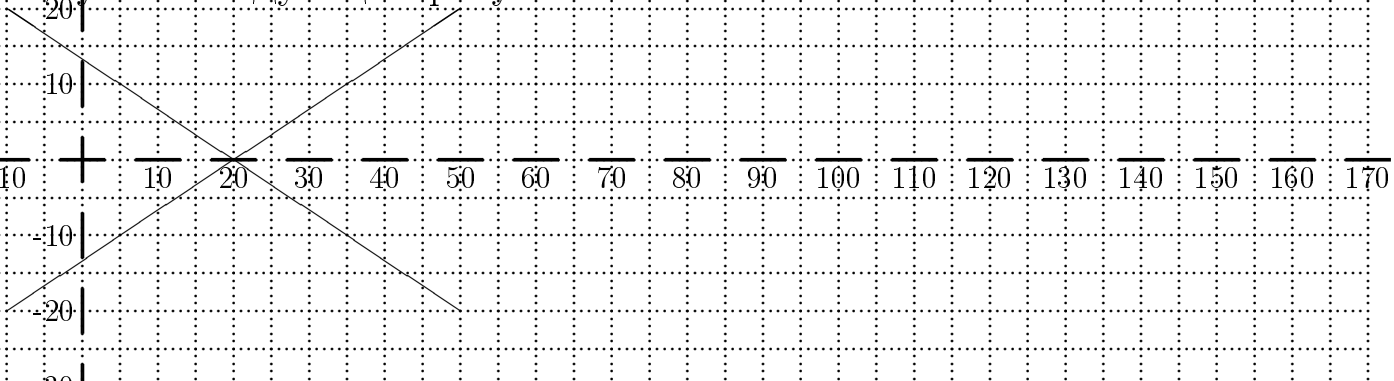
получим следующий результат:

\setka — команда для изображения координатной сетки в окружении picture.

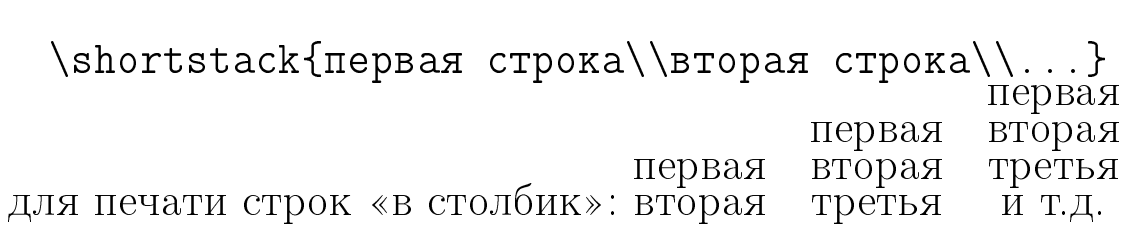
Например, в результате компиляции фрагмента

```
{\unitlength=1mm
\begin{picture}(60,40)(-10,-20)
  \setka
  \put(-10.00,-20.00){\line ( 3, 2){060.00}}
  \put(050.00,-20.00){\line (-3, 2){060.00}}
\end{picture}}
```

получим следующий результат:



`\shortstack{первая строка\\вторая строка\\...}` — команда для печати строк «в столбик»:



`\small` — команда переключения размера шрифта.

Например, {\Huge Самый большой}, {\huge поменьше}, {\LARGE ещё меньше}, {\Large ещё меньше}, {\large ещё меньше}, {\normalsize ещё меньше}, {\small ещё меньше}, {\tiny совсем мелко}.

`sogl` — имя окружения для оформления соглашений, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании **`\newtheorem`**.

Как для любого другого окружения, с окружением `sogl` связан одноименный счетчик, причем после выполнения `\begin{sogl}` значение счетчика `sogl` увеличивается на 1.

`sogl` — имя окружения для оформления соглашений, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды \ThmEnvCom.

Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#sogl# равно~\arabic{sogl}.
 \begin{sogl}
В равенстве \mbox{$L(x)=R(x)$}, справедливом для всех значений
аргумента~$x$, вместо~$=$ используется символ~$\equiv$.
 \end{sogl}
Теперь значение счетчика~\verb#sogl# равно~\arabic{sogl}.
```

дает следующий результат:

Начальное значение счетчика `sogl` равно 0.

Соглашение 1. *В равенстве $L(x) = R(x)$, справедливом для всех значений аргумента x, вместо $=$ используется символ $\equiv$.*

Теперь значение счетчика `sogl` равно 1.

`\sqrt{...}` — команда для печати знака квадратного корня (работает в математической моде.

Например, в результате компиляции выражения
`$2+\sqrt{a+1}$` получим $2+\sqrt{a+1}$.

`\stackrel{`u`}{`s`}` — команда для вставки символа u (уменьшенным шрифтом, как для индекса) над символом s. Работает только в математической моде!

Например, для кода `$\stackrel{x}{\to}$` получаем такой результат компиляции: $\stackrel{x}{\to}$.

`\strut` — команда вставки в текст «невидимой линейки», т.е. прямоугольника нулевой ширины, причём
выступающий вниз на уровень буквы с самым «низким хвостом»: q,у,ф...
и выступающий вверх на уровень буквы с самым «высоким ухом»: h,б...

`\symbol{код}` — команда печати символа с данным кодом.

Имеем \symbol{1}´; \symbol{2}ˆ; \symbol{3}˜; \symbol{4}¨; \symbol{5}˝; \symbol{6}˚; \symbol{7}ˇ; \symbol{8}˘; \symbol{9}¯; \symbol{10}˙; \symbol{11}¸; \symbol{12}˛; \symbol{13}I; \symbol{14}⟨; \symbol{15}⟩; \symbol{16}“; \symbol{17}”; \symbol{18}‘; \symbol{19}’; \symbol{20}‚; \symbol{21}–; \symbol{22}—; \symbol{23}; \symbol{24}₀; \symbol{25}ı; \symbol{26}ȷ; \symbol{27}ff; \symbol{28}fi; \symbol{29}fl; \symbol{30}ffi; \symbol{31}ffl; \symbol{32}␣; \symbol{33}!; \symbol{34}"; \symbol{35}#; \symbol{36}$; \symbol{37}%; \symbol{38}&; \symbol{39}’; \symbol{40}(; \symbol{41}); \symbol{42}*; \symbol{43}+; \symbol{44},; \symbol{45}-; \symbol{46}.; \symbol{47}/; \symbol{48}0; \symbol{49}1; \symbol{50}2; \symbol{51}3; \symbol{52}4; \symbol{53}5; \symbol{54}6; \symbol{55}7; \symbol{56}8; \symbol{57}9; \symbol{58}:; \symbol{59};; \symbol{60}<; \symbol{61}=; \symbol{62}>; \symbol{63}?; \symbol{64}@; \symbol{65}A;

\symbol{66}B; \symbol{67}C; \symbol{68}D; \symbol{69}E;
\symbol{70}F; \symbol{71}G; \symbol{72}H; \symbol{73}I; \symbol{74}J;
\symbol{75}K; \symbol{76}L; \symbol{77}M; \symbol{78}N;
\symbol{79}O; \symbol{80}P; \symbol{81}Q; \symbol{82}R;
\symbol{83}S; \symbol{84}T; \symbol{85}U; \symbol{86}V;
\symbol{87}W; \symbol{88}X; \symbol{89}Y; \symbol{90}Z;
\symbol{91}[; \symbol{92}\; \symbol{93}]; \symbol{94}^; \symbol{95}_;
\symbol{96}‘; \symbol{97}a; \symbol{98}b; \symbol{99}c; \symbol{100}d;
\symbol{101}e; \symbol{102}f; \symbol{103}g; \symbol{104}h;
\symbol{105}i; \symbol{106}j; \symbol{107}k; \symbol{108}l;
\symbol{109}m; \symbol{110}n; \symbol{111}o; \symbol{112}p;
\symbol{113}q; \symbol{114}r; \symbol{115}s; \symbol{116}t;
\symbol{117}u; \symbol{118}v; \symbol{119}w; \symbol{120}x;
\symbol{121}y; \symbol{122}z; \symbol{123}{; \symbol{124}|;
\symbol{125}}; \symbol{126}~; \symbol{127}-; \symbol{128}Γ;
\symbol{129}Ғ; \symbol{130}Ъ; \symbol{131}Ħ; \symbol{132}Ħ;

\symbol{133}Җ; \symbol{134}Ҙ; \symbol{135}Љ; \symbol{136}Ї;
\symbol{137}Ҟ; \symbol{138}Ҡ; \symbol{139}Ӄ; \symbol{140}Æ;
\symbol{141}Ң; \symbol{142}Ҥ; \symbol{143}S; \symbol{144}Ө;
\symbol{145}Ҫ; \symbol{146}Ў; \symbol{147}Y; \symbol{148}Ұ;
\symbol{149}Ҳ; \symbol{150}Ҷ; \symbol{151}Ч; \symbol{152}Ҷ;
\symbol{153}Є; \symbol{154}Ә; \symbol{155}Њ; \symbol{156}Ё;
\symbol{157}№; \symbol{158}¤; \symbol{159}§; \symbol{160}ґ;
\symbol{161}г; \symbol{162}ђ; \symbol{163}ћ; \symbol{164}h;
\symbol{165}ж; \symbol{166}ҙ; \symbol{167}љ; \symbol{168}ї;
\symbol{169}к; \symbol{170}ҡ; \symbol{171}ӄ; \symbol{172}æ;
\symbol{173}ң; \symbol{174}ҥ; \symbol{175}s; \symbol{176}ө;
\symbol{177}ҫ; \symbol{178}ў; \symbol{179}ү; \symbol{180}ұ;
\symbol{181}ҳ; \symbol{182}ҷ; \symbol{183}ч; \symbol{184}ҷ;
\symbol{185}є; \symbol{186}ә; \symbol{187}њ; \symbol{188}ё;
\symbol{189}„; \symbol{190}«; \symbol{191}»; \symbol{192}A;

\symbol{193}Б; \symbol{194}В; \symbol{195}Г; \symbol{196}Д;
\symbol{197}Е; \symbol{198}Ж; \symbol{199}З; \symbol{200}И;
\symbol{201}Й; \symbol{202}К; \symbol{203}Л; \symbol{204}М;
\symbol{205}Н; \symbol{206}О; \symbol{207}П; \symbol{208}Р;
\symbol{209}С; \symbol{210}Т; \symbol{211}У; \symbol{212}Ф;
\symbol{213}Х; \symbol{214}Ц; \symbol{215}Ч; \symbol{216}Ш;
\symbol{217}Щ; \symbol{218}Ъ; \symbol{219}Ы; \symbol{220}Ь;
\symbol{221}Э; \symbol{222}Ю; \symbol{223}Я; \symbol{224}а;
\symbol{225}б; \symbol{226}в; \symbol{227}г; \symbol{228}д;
\symbol{229}е; \symbol{230}ж; \symbol{231}з; \symbol{232}и;
\symbol{233}й; \symbol{234}к; \symbol{235}л; \symbol{236}м;
\symbol{237}н; \symbol{238}о; \symbol{239}п; \symbol{240}р;
\symbol{241}с; \symbol{242}т; \symbol{243}у; \symbol{244}ф;
\symbol{245}х; \symbol{246}ц; \symbol{247}ч; \symbol{248}ш;
\symbol{249}щ; \symbol{250}ъ; \symbol{251}ы; \symbol{252}ь;
\symbol{253}э; \symbol{254}ю; \symbol{255}я;

T

`\tableofcontents` — команда вставки оглавления или описания содержания.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

В данном случае у окружения `tabular` имеется 4 параметра, что означает, что в таблице должно быть 4 столбца.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

Параметр p{*длина*} означает, что элементы соответствующего столбца (в данном случае первого) форматируются в абзац шириной *длина*.

`tabular` — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

Символы | слева и справа от `p{`*длина*`}` означают, что этот столбец слева и справа окружён границами, причём справа — двойной границей.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

Символы | слева и справа от p{*длина*} означают, что этот столбец слева и справа окружён границами, причём справа — двойной границей.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

Символы | слева и справа от p{*длина*} означают, что этот столбец слева и справа окружён границами, причём справа — двойной границей.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

r — сокращение от right.

Символ r означает, что элементы этого столбца (в данном случае второго) выравниваются по правому краю.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

l — сокращение от left.

Символ l означает, что элементы этого столбца (в данном случае третьего) выравниваются по левому краю.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

c — сокращение от center.

Символ c означает, что элементы этого столбца (в данном случае четвёртого) выравниваются по центру.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

Команда multicolumn предназначена для выделения элемента строки, занимающего несколько столбцов.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

Первый параметр 3 команды `multicolumn` означает, что этот элемент занимает 3 столбца.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

Второй параметр «c» команды multicolumn означает, что этот элемент центрируется.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

Вертикальные скобки «|c|» у второго параметра команды multicolumn означают, что этот элемент слева и справа будет окружён вертикальными границами.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

В строке элементы разных столбцов разделены символами &.

tabular — окружение для **создания таблиц** в текстовой моде.

```
\begin{tabular}{|p{0.14\textwidth}||rlc}
    \hline
  Абзац в несколько строк
    & 2 & 3 & 4\\
    \hline
  a & b & c & d\\
  123 & 234 & 345 & 456\\
    \cline{3-4}
  \multicolumn{3}{|c|}{Длинный
элемент}  & S\\
    \hline
\end{tabular}
```

Абзац в несколько строк	2	3	4
a	b	c	d
123	234	345	456
Длинный элемент			S

Каждая строка таблицы завершается командой \\.

`\the`*идентификатор* — команда печати значения регистра или счётчика.

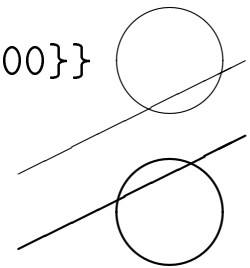
Например, в результате выполнения кода

```
\newcount\Fgh  \newcounter{Abcd}
\Fgh=4
\setcounter{Abcd}{5}
\the\Fgh, \theAbcd
```

получим: 4, 5.

`\thicklines` — команда, применяемая в окружении `\picture` для некоторого увеличения толщины линий.

```
\put(-15.00,-05.00){\line( 2, 1){040.00}}
\put(-15.00,-15.00){\thicklines\line( 2, 1){040.00}}
\put(005.00,010.00){\circle{020.00}}
\put(005.00,-10.00){\thicklines\circle{020.00}}
```



`\tiny` — команда переключения на очень мелкий размер шрифта. Например, {\Huge Самый большой}, {\huge поменьше}, {\LARGE ещё меньше}, {\Large ещё меньше}, {\large ещё меньше}, {\normalsize ещё меньше}, {\small ещё меньше}, {\tiny совсем мелко}.

`thm` — имя окружения для оформления теорем, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании **`\newtheorem`**.

Как для любого другого окружения, с окружением `thm` связан одноименный счетчик, причем после выполнения `\begin{thm}` значение счетчика `thm` увеличивается на 1.

`thm` — имя окружения для оформления теорем, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды \ThmEnvCom.

Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#thm# равно~\arabic{thm}.
 \begin{thm}
Если треугольник является равнобедренным, то
медиана, биссектриса и высота, опущенные на его основание, совпадают.
 \end{thm}
Теперь значение счетчика~\verb#thm# равно~\arabic{thm}.
```

дает следующий результат:

Начальное значение счетчика `thm` равно 0.

Теорема 1. *Если треугольник является равнобедренным, то медиана, биссектриса и высота, опущенные на его основание, совпадают.*

Теперь значение счетчика `thm` равно 1.

`\ThmEnvCom` — **команда для автоматизации** оформления окружений **defnt**, **lmm**, **thm**, **zam**, **cons**, **aim**, **quest**.

Каждому из этих окружений соответствует одноимённый **счётчик**, т.е. определены **счётчики** `defnt`, `lmm` и др.

`\ThmEnvEquat{арг1}{арг2}{арг3}{арг4}` — команда для введения нумерованных формул в определениях, теоремах и др., сформированных с помощью `\ThmEnvCom`. Например, в результате выполнения `\ThmEnvCom{defnt}{AAA}{Формула в тексте определения:`

```
\ThmEnvEquat{#1}{#2}{EquatdefntAAA}{\ln{x^{3}}=3\ln{x},}
\newline продолжение текста определения.}
```

будет определена команда `\ComTextdefntAAA{}{}{}{}{}{}`.

При выполнении `\ComTextdefntAAA{#1}{0}{3}{4}{5}{6}` будет напечатан текст определения, увеличены счетчики `defnt` и `equation`, новое значение счетчика `equation` будет присвоено метке `EquatdefntAAA` и этим значением счетчика будет отмечена формула $\ln x^3 = 3 \ln x$.

В дальнейшем следует применять `\ComTextdefntAAA{#1}{1}{3}{4}{5}{6}`.

`\thesection`, `\thesubsection`, `\thesubsubsection` — команды распечатки номера раздела (`\section`), подраздела (`\subsection`) и, соответственно, (`\subsubsection`).

`\TextPict` — команда вывода текста слева от рисунка, определённая в `texsample.tex`

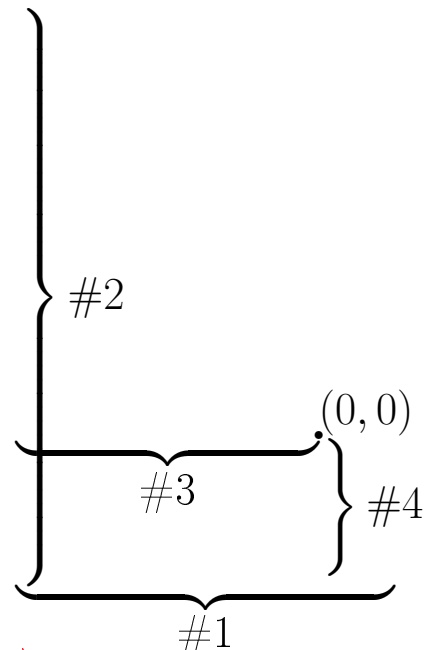
Имеет 9 обязательных параметров:

#1 — натуральное число, означающее ширину рисунка в миллиметрах;

#2 — натуральное число, означающее высоту рисунка в миллиметрах;

#3 — натуральное число, означающее смещение вправо точки с координатами $(0,0)$ относительно левого нижнего края рисунка;

#4 — натуральное число, означающее смещение вверх точки с координатами $(0,0)$ относительно левого нижнего края рисунка;



#5 — система команд для окружения **picture**: **\put**... и др.;

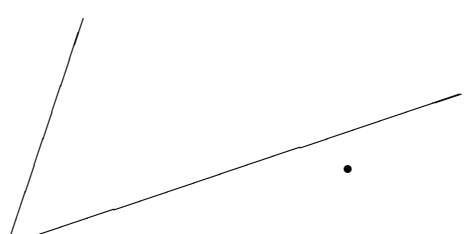
#6 — текст, печатаемый справа от рисунка, #7,#8,#9 — свободны.

Для вывода текста справа от рисунка применяется **PictText**.

`\TextPict` — команда вывода текста слева от рисунка.

```
\TextPict{60}{30}{45}{10}{
 \put(-45.00,-10.00){\line( 3, 1){060.00}}
 \put(-45.00,-10.00){\line( 1, 3){010.00}}
 \put(000.00,000.00){\circle*{1}}
}{Текст может содержать несколько абзацев.\par
  Использованы команды
  \symbol{92}put, \symbol{92}line, \symbol{92}circle,
  \symbol{92}symbol.
}{}{}{}
```

Текст может содержать несколько абзацев.
Использованы команды
\put, \line, \circle, \symbol.



Текстовая мода

`\textbf{...}` — команда перехода к **жирному шрифту**.

`\textit{...}` — команда перехода к *курсиву*.

`\textrm{...}` — команда перехода к шрифту с засечками.

`\textsf{...}` — команда перехода к прямому шрифту.

`\textsl{...}` — команда перехода к наклонному шрифту.

`\texttt{...}` — команда перехода к прямому моноширинному шрифту.

Математическая мода

`\mathbf{...}` — команда перехода к **жирному шрифту**.

`\mathit{...}` — команда перехода к *курсиву*.

`\mathrm{...}` — команда перехода к шрифту с засечками.

`\mathbb{...}` — команда перехода к «пустотелому» шрифту $\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$.

`\textheight` — это команда, один из параметров макета страницы, а именно `\textheight` равно значение высоты прямоугольника, отведенного на макете страницы под текст.

`\textheight` — это команда, один из параметров макета страницы, а именно `\textheight` равно значение высоты прямоугольника, отведенного на макете страницы под текст.

`\textwidth` — это команда, один из параметров макета страницы, а именно `\textwidth` равно ширине прямоугольника, отведенного на макете страницы под текст.

`\typeout{текст, выводимый в *.log}` — команда для вывода некоторого текста в файл `*.log`.

`\TwoBlock{...}{...}{...}{...}{...}` — команда, **определённая** в `texsample.tex`, обеспечивающая вывод информации в виде двух горизонтальных блоков одинаковой высоты (см. `fTwoBlock`).

Первый и второй аргументы `#1` и `#2` означают ширину первого и, соответсвенно, второго блока, измеряемую в долях от ширины текста, т.е. от значения переменной `\textwidth`.

Третий аргумент `#3` равен высоте блоков (высота у них будет одинаковой) в тех же единицах.

Значением четвертого `#4` и пятого аргумента `#5` являются содержание первого и, соответственно, второго блока. При этом текст автоматически форматируется по ширине столбца.

`\TwoBlock{...}{...}{...}{...}{...}` — команда, **определённая** в `texsample.tex`, обеспечивающая вывод информации в виде двух горизонтальных блоков одинаковой высоты (см. `fTwoBlock`).

Компиляция фрагмента

```
\TwoBlock{0.2}{0.25}{0.1} {Первая колонка текста}
{Вторая колонка текста.}
```

даёт следующий результат:

Первая ко- Вторая колонка
лонка текста текста

`\TwoBlock{...}{...}{...}{...}{...}` — команда, **определённая** в `texsample.tex`, обеспечивающая вывод информации в виде двух горизонтальных блоков одинаковой высоты (см. `fTwoBlock`).

Компиляция фрагмента

```
\TwoBlock{0.4}{0.45}{0.2} {\color{magenta}Первая колонка,
шириной \mbox{\texttt{0.4\symbol{92}textwidth}}, высотой
\mbox{\texttt{0.2\symbol92textwidth}.}
{Вторая колонка, шириной
\mbox{\texttt{0.45\symbol{92}textwidth}}, высотой \mbox{\text
```

даёт следующий результат:

Первая колонка, шириной 0.4\textwidth, высотой 0.2\textwidth.

Вторая колонка, шириной 0.45\textwidth, высотой 0.2\textwidth.

U

`\underbrace{`*формула1*`}_{`*формула2*`}` — команда, действующая в математической моде, ставящая ***под*** «*формула1*» фигурную скобку, завершаюся снизу выражением «*формула2*».

Часть этой команды вида `_{`*формула2*`}` является необязательной.

Например, код

`$x+\underbrace{y+z}+t$` и `$x+\underbrace{y+z}_{\alpha}+t$`

в результате компиляции даёт такой результат:

$x+\underbrace{y+z}+t$ и $x+\underbrace{y+z}_{\alpha}+t$

`\underline{`подчёркнутый текст`}` — команда, формирующая из строки «`подчёркнутый текст`» бокс и подчёркивающий его. Работает и в текстовой, и в математической модах.

Например, компиляция кода

```
«В этом \underline{тексте} часть \underline{слов
подчёркнута.} При этом \underline{перенос подчёрнутого
текста на новую строку} не производится.»
```

приводит к такому результату:

В этом тексте часть слов подчёркнута. При этом перенос подчёрнутого не производится.

`\underline{`подчёркнутый текст`}` — команда, формирующая из строки «`подчёркнутый текст`» бокс и подчёркивающий его. Работает и в текстовой, и в математической модах.

Например, компиляция кода

```
«В этом \underline{тексте} часть \underline{слов
подчёркнута.} При этом \underline{перенос подчёрнутого
текста на новую строку} не производится.»
```

приводит к такому результату:

В этом тексте часть слов подчёркнута. При этом перенос подчёрнутого не производится.

```
\underline{Линия} \underline{подчёркивания}.
```

Линия подчёркивания.

Неприятно, когда линия подчёркивания в разных словах проходит на разной высоте. Это можно устранить с помощью `\strut`.

`\underline{`подчёркнутый текст`}` — команда, формирующая из строки «`подчёркнутый текст`» бокс и подчёркивающий его. Работает и в текстовой, и в математической модах.

Например, компиляция кода

```
«В этом \underline{тексте} часть \underline{слов
подчёркнута.} При этом \underline{перенос подчёрнутого
текста на новую строку} не производится.»
```

приводит к такому результату:

В этом тексте часть слов подчёркнута. При этом перенос подчёрнутого не производится.

```
\underline{Линия\strut} \underline{подчёркивания\strut}.
```

Линия подчёркивания.

Неприятно, когда линия подчёркивания в разных словах проходит на разной высоте. Это можно устранить с помощью `\strut`.

`\unitlength=a` — команда присвоения переменной `\unitlength` значения единицы длины. У единицы длины a обязательно должна быть указана размерность: `pt`, `mm`, `cm`. Так, например, `\unitlength=10mm` делает единицу длины равной 10 миллиметров, `\unitlength=0.3cm` делает единицу длины равной 0,3 сантиметра.

Указание единицы длины является существенным при использовании, например, команд `\put`, `\`

V

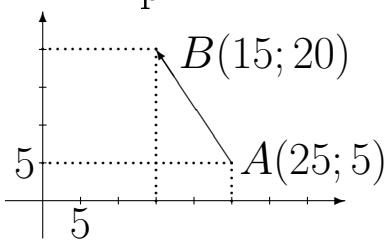
`\vector(a,b){c}` — команда рисования направленного отрезка (стрелки) в окружении `picture`. Формат команды почти повторяет формат команды `\line`. Здесь также c — это проекция изображаемого отрезка на ось абсцисс, если отрезок расположен наклонно или ортогонально по отношению к оси ординат. Если же этот отрезок параллелен оси ординат, то c равен проекции изображаемого отрезка на *ось ординат*. В отличие от команды `\line` абсолютное значение целых чисел a, b не превосходит 4 (а не 6, как в команде `\line`) то есть $\begin{cases} -4 \le a \le 4, \\ -4 \le b \le 4. \end{cases}$

Как и для `\line`, если a и b не равны 0, числа a и b должны быть взаимно простыми числами.

`\vector(a,b){c}` — команда рисования направленного отрезка (стрелки) в окружении `picture`. Формат команды почти повторяет формат команды `\line`.

Например, команда

`\put(25.00,05.00){\vector(-2,3){10}}` изобразит направленный отрезок AB:



`\vec{`буква`}` — команда для постановки стрелки над буквой. Работает в математической моде. См. также `\vectr`.

Например, результатом компиляции кода

`$\vec{a}$` будет $\vec{a}$,

`$\vec{p}$` будет $\vec{p}$,

`$\vec{\alpha}$` будет $\vec{\alpha}$.

`\vectr{`буква`}` — команда для постановки стрелки над буквой, причём буква печатается в полужирным шрифтом `\mathbf`. Работает в математической моде. См. также `\vec`.

Например, результатом компиляции кода
`$\vectr{a}$` будет $\overrightarrow{\mathbf{a}}$,
`$\vectr{p}$` будет $\overrightarrow{\mathbf{p}}$,
`$\vectr{\alpha}$` будет $\overrightarrow{\alpha}$.

`vmatrix` — окружение для матрицы в вертикальных скобках.

В результате компиляции кода

```
$\begin{vmatrix}
1 & 2\\ 3 & 4\\
\end{vmatrix},\quad
\begin{vmatrix}
1 & 2\\ 3 & 4\\ 5 & 6\\
\end{vmatrix},\quad
\begin{vmatrix}
1 & 2 & 3\\ 4 & 5 & 6\\
\end{vmatrix}.$
```

получим

$$\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}, \quad \begin{vmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{vmatrix}, \quad \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{vmatrix}.$$

`\vspace{`h`}` — команда вставки дополнительного расстояния между абзацами, равного h.

Указание **единиц измерения** величины h обязательно!

Например, перед этим абзацем была вставлена команда `\vspace{20mm}`

W

`white` — белый цвет. Используется, например в составе команды

```
{\color{white}белый цвет}.
```

`\widehat` — команда для постановки знака $\widehat{}$ над словом, в математической моде. Например, результат компиляции следующего фрагмента

```
$a,\widehat{a};\quad ab,\widehat{ab};\quad abc,\widehat{abc}$
```

дает следующий результат:

$a,\widehat{a};\quad ab,\widehat{ab};\quad abc,\widehat{abc}$

X

Y

`yellow` — голубой цвет. Используется, например в составе команды `{\color{yellow}ж\symbol{188}лтый цвет}`.

Z

`zad` — имя окружения для оформления задач, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`, а для задач — `\MakExecisePdf`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании **`\newtheorem`**.

Как для любого другого окружения, с окружением `zad` связан одноименный счетчик, причем после выполнения `\begin{zad}` значение счетчика `zad` увеличивается на 1.

`zad` — имя окружения для оформления задач, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`, а для задач — `\MakExecisePdf`.

Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#zad# равно~\arabic{zad}.
 \begin{zad}
Вычислите $\arcsin{0,8}+\arcsin{0,6}$.
 \end{zad}
Теперь значение счетчика~\verb#zad# равно~\arabic{zad}.
```

дает следующий результат:

Начальное значение счетчика `zad` равно 0.

Задача 1. *Вычислите* $\arcsin 0,8 + \arcsin 0,6$.

Теперь значение счетчика `zad` равно 1.

`zam` — имя окружения для оформления замечаний, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды `\ThmEnvCom`.

Остальные окружения типа «теорема», введённый в `texsample.tex`, перечислены в описании **`\newtheorem`**.

Как для любого другого окружения, с окружением `zam` связан одноименный счетчик, причем после выполнения `\begin{zam}` значение счетчика `zam` увеличивается на 1.

`zam` — имя окружения для оформления замечаний, определенное в файле `texsample.tex`. Автоматизировать создание окружений типа «теорема» можно с помощью команды \ThmEnvCom.

Например, компиляция фрагмента

```
Начальное значение счетчика~\verb#zam# равно~\arabic{zam}.
 \begin{zam}
Равенство двух углов треугольника равносильно
равенству длин его двух сторон.
 \end{zam}
Теперь значение счетчика~\verb#zam# равно~\arabic{zam}.
```

дает следующий результат:

Начальное значение счетчика `zam` равно 0.

Замечание 1. *Равенство двух углов треугольника равносильно равенству длин его двух сторон.*

Теперь значение счетчика `zam` равно 1.

IV. Команды `testsample.tex`

`\coeff{`n`}` — команда для создания поля, предназначенного для ввода числа, где n — это число, верный ответ к заданию.

Например, команда `\coeff{5}` создаст квадратик, в который надо ввести 5, чтобы получить балл за правильный ответ.

`\ComAdjMatrB{`a_1`}{`a_2`}{`a_3`}...{`a_9`}` — команда для вычисления коэффициентов матрицы $\begin{pmatrix} \texttt{countVAx} & \texttt{countVAy} & \texttt{countVAz} \\ \texttt{countVBx} & \texttt{countVBy} & \texttt{countVBz} \\ \texttt{countVCx} & \texttt{countVCy} & \texttt{countVCz} \end{pmatrix}$ (указаны имена **счетчиков**, в которых хранятся коэффициенты), состоящей из алгебраических дополнений к элементам матриц $\begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix}$. Иными словами, матрица $\begin{pmatrix} \texttt{countVAx} & \texttt{countVAy} & \texttt{countVAz} \\ \texttt{countVBx} & \texttt{countVBy} & \texttt{countVBz} \\ \texttt{countVCx} & \texttt{countVCy} & \texttt{countVCz} \end{pmatrix}$ является транспонированной к матрице, присоединенной к $\begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix}$.

\ComCypherOfNumbA{$Name$}{k}{n} — команда для вычисления цифр числа n. Здесь k — количество знаков числа, команда определяет команды

\Namea,\Nameb,...,\Nameбуква$_k$

$n = \mathtt{\backslash namea} \cdot 10^{k-1} + \overline{\mathtt{\backslash nameb} \cdot 10^{k-2}} + \mathtt{\backslash namea} \cdot 10^{k-3} + \ldots$

Выполнение команды \ComCypherOfNumbA{Pq}{3}{936} приводит к появлению команд:

\Pqa, равное 9; \Pqb, равное 3; \Pqc, равное 6.

Реально число цифр может быть другим:

\ComCypherOfNumbA{Nmb}{3}{27} приводит к появлению команд:

\Nmba, равное 0, \Nmbb, равное 2, \Nmbc, равное 7;

\ComCypherOfNumbA{Qw}{2}{362} приводит к появлению команд:

\Qwa, равное 6, \Qwb, равное 2.

`\ComDetTwo{a}{b}{c}{d}` — команда для вычисления детерминанта матрицы $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$.

Значение детерминанта присваивается **регистру** `\CountSca`.

Например, компиляция кода

```
$\det\begin{pmatrix}
        2 & 3\\
        5 & 4\\
      \end{pmatrix}=
\ComDetTwo{2}{3}{5}{4}
\the\CountSca.$
```

приводит к следующему результату:

$\det\begin{pmatrix} 2 & 3 \\ 5 & 4 \end{pmatrix} = -7.$

`\ComDetThree{`a_{11}`}{`a_{12}`}{`a_{13}`}{`a_{21}`}{`a_{22}`}{`a_{23}`}{`a_{31}`}{`a_{32}`}{`a_{33}`}` — команда для вычисления детерминанта матрицы $\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$.

Значение детерминанта присваивается **регистру** `\CountSca`.

Например, компиляция кода

```
$\begin{vmatrix}
    1 & 2 & 3\\
    4 & 5 & 6\\
    7 & 8 & 9\\
 \end{vmatrix}=
\ComDetThree{1}{2}{3}{4}{5}{6}{7}{8}{9}
\the\CountSca.$
```

приводит к следующему результату: $\det \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 6 \\ 7 & 8 & 9 \end{pmatrix} = 0.$

\ComModbyNumbSingleNonZero{n}{m}{k}:

для $n > 0$ \CountTmp $= m - p \cdot n + k$, где $0 \leqslant m - n \cdot p \leqslant n - 1$,

для $n < 0$ \CountTmp $= p \cdot n - m + k$, где $0 \leqslant n \cdot p - m \leqslant n - 1$,

т.е. \CountTmp $\in [k;\ m + k) = [k;\ m + k - 1]$.

\ComModbyNumbSingleNonZero{12}{3}{0}, \CountTmp=0.

\ComModbyNumbSingleNonZero{12}{3}{1}, \CountTmp=1.

\ComModbyNumbSingleNonZero{12}{3}{2}, \CountTmp=2.

\ComModbyNumbSingleNonZero{12}{4}{0}, \CountTmp=0.

\ComModbyNumbSingleNonZero{12}{4}{1}, \CountTmp=1.

\ComModbyNumbSingleNonZero{12}{4}{2}, \CountTmp=2.

\ComModbyNumbSingleNonZero{12}{5}{0}, \CountTmp=2.

\ComModbyNumbSingleNonZero{12}{5}{1}, \CountTmp=3.

\ComModbyNumbSingleNonZero{12}{5}{2}, \CountTmp=4.

\ComModbyNumbDoubleNonZero{n}{m}{k} — команда для вычисления числа в регистре \CountTmp:

$\backslash\mathtt{CountTmp} \in (-m-k;\ -k] \cup [k;\ m+k) =$

$= (-m-k+1;\ -k] \cup [k;\ m+k-1].$

\ComModbyNumbSingleNonZero{12}{3}{0}, \CountTmp=0.

\ComModbyNumbSingleNonZero{12}{3}{1}, \CountTmp=1.

\ComModbyNumbSingleNonZero{12}{3}{2}, \CountTmp=2.

\ComModbyNumbSingleNonZero{12}{5}{0}, \CountTmp=2.

\ComModbyNumbSingleNonZero{12}{5}{1}, \CountTmp=3.

\ComModbyNumbSingleNonZero{12}{5}{2}, \CountTmp=4.

\ComModbyNumbSingleNonZero{-12}{5}{0}, \CountTmp=2.

\ComModbyNumbSingleNonZero{-12}{5}{1}, \CountTmp=3.

\ComModbyNumbSingleNonZero{-12}{5}{2}, \CountTmp=4.

\ComModbyNumbSingleNonZero{-12}{5}{3}, \CountTmp=5.

\ComModbyNumbSingleNonZero{-12}{5}{4}, \CountTmp=6.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения
`\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{3}{x}\ComPlusL{1}{-4}{}`
будет

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения
`\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{3}{x}\ComPlusL{1}{-4}{}`
будет
$2x^2+3x-4$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения

```
\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{3}{x}\ComPlusL{1}{-4}{}
```

будет

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения

```
\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{3}{x}\ComPlusL{1}{-4}{}
```

будет

$+3x-4$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения
`\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{1}{-4}{}`
будет

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения
`\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{1}{-4}{}`
будет
-4.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения
`\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{1}{0}{}`
будет

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения
`\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{1}{0}{}`
будет
. (Т.е. ничего не напечатано).

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения
`\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{3}{0}{}`
будет

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

При $k = 0$ выражение $0e$ не печатается ни при каком значении i, кроме случая, когда $i = 3$ и все предыдущие двучлены в печатаемом выражении были нулевыми.

Например, результатом компиляции выражения
`\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{3}{0}{}`
будет
0.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$:

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$: во-первых, при $k > 0$ знак $+$ не ставится;

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$: во-первых, при $k > 0$ знак $+$ не ставится;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$: во-первых, при $k > 0$ знак $+$ не ставится;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

$2x^2+2x+3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$: во-первых, при $k > 0$ знак $+$ не ставится;

```
\ComPlusL{0}{-2}{x^{2}}\ComPlusL{1}{-2}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$: во-первых, при $k > 0$ знак $+$ не ставится;

```
\ComPlusL{0}{-2}{x^{2}}\ComPlusL{1}{-2}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

$-2x^2-2x+3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$:
во-первых, при $k > 0$ знак $+$ не ставится;
во-вторых, при $k = 1$ вместо $1e$ печатается e;

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$:
во-первых, при $k > 0$ знак $+$ не ставится;
во-вторых, при $k = 1$ вместо $1e$ печатается e;

```
\ComPlusL{0}{1}{x^{2}}\ComPlusL{1}{1}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$:
во-первых, при $k > 0$ знак $+$ не ставится;
во-вторых, при $k = 1$ вместо $1e$ печатается e;

```
\ComPlusL{0}{1}{x^{2}}\ComPlusL{1}{1}{x}\ComPlusL{1}{3}{}
```
при компиляции даёт
x^2+x+3.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$:
во-первых, при $k > 0$ знак $+$ не ставится;
во-вторых, при $k = 1$ вместо $1e$ печатается e;
в-третьих, при $k = -1$ вместо $-1e$ печатается $-e$;

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$:
во-первых, при $k > 0$ знак $+$ не ставится;
во-вторых, при $k = 1$ вместо $1e$ печатается e;
в-третьих, при $k = -1$ вместо $-1e$ печатается $-e$;
`\ComPlusL{0}{-1}{x^{2}}\ComPlusL{1}{-1}{x}\ComPlusL{1}{3}{}`
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 0$ применяется для печати первого двучлена $k \cdot e$:
во-первых, при $k > 0$ знак $+$ не ставится;
во-вторых, при $k = 1$ вместо $1e$ печатается e;
в-третьих, при $k = -1$ вместо $-1e$ печатается $-e$;
`\ComPlusL{0}{-1}{x^{2}}\ComPlusL{1}{-1}{x}\ComPlusL{1}{3}{}`
при компиляции даёт
$-x^2-x+3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

$2x^2+2x+3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;

```
\ComPlusL{0}{-2}{x^{2}}\ComPlusL{1}{-2}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;

```
\ComPlusL{0}{-2}{x^{2}}\ComPlusL{1}{-2}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт
$-2x^2-2x+3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;
во-вторых, при $k = 1$ вместо $1e$ печатается $+e$;

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;
во-вторых, при $k = 1$ вместо $1e$ печатается $+e$;

```
\ComPlusL{0}{1}{x^{2}}\ComPlusL{1}{1}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;
во-вторых, при $k = 1$ вместо $1e$ печатается $+e$;

```
\ComPlusL{0}{1}{x^{2}}\ComPlusL{1}{1}{x}\ComPlusL{1}{3}{}
```

при компиляции даёт

x^2+x+3.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;
во-вторых, при $k = 1$ вместо $1e$ печатается $+e$;
в-третьих, при $k = -1$ вместо $-1e$ печатается $-e$;

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;
во-вторых, при $k = 1$ вместо $1e$ печатается $+e$;
в-третьих, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{-1}{x^{2}}\ComPlusL{1}{-1}{x}\ComPlusL{1}{3}{}
```
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;
во-вторых, при $k = 1$ вместо $1e$ печатается $+e$;
в-третьих, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{-1}{x^{2}}\ComPlusL{1}{-1}{x}\ComPlusL{1}{3}{}
```
при компиляции даёт
$-x^2-x+3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;
во-вторых, при $k = 1$ вместо $1e$ печатается $+e$;
в-третьих, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{1}{3}{}
```
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 1$ применяется для печати двучлена $k \cdot e$, который может оказаться не первым ненулевым:
во-первых, при $k \geqslant 2$ вместо ke печатается $+ke$;
во-вторых, при $k = 1$ вместо $1e$ печатается $+e$;
в-третьих, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{1}{3}{}
```
при компиляции даёт
$+3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{3}{}
```

при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{3}{}
```

при компиляции даёт
$2x^2+2x+3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{-3}{}
```

при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{-3}{}
```

при компиляции даёт
$2x^2+2x-3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{1}{}
```

при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{1}{}
```

при компиляции даёт
$2x^2+2x+1$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{-1}{}
```
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{-1}{}
```
при компиляции даёт
$2x^2+2x-1$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;
в-третьих, при $k = 0$ ничего не печатается.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;
в-третьих, при $k = 0$ ничего не печатается.
`\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{0}{}`
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 2$ применяется для печати двучлена $k \cdot e$ при пустом e, т.е. печати k:
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;
в-третьих, при $k = 0$ ничего не печатается.
`\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{2}{0}{}`
при компиляции даёт
$2x^2+2x$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{3}{3}{}
```
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{3}{3}{}
```
при компиляции даёт
$2x^2+2x+3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{3}{-3}{}
```
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{3}{-3}{}
```
при компиляции даёт
$2x^2+2x-3$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{3}{1}{}
```
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{2}{x}\ComPlusL{3}{1}{}
```
при компиляции даёт
$2x^{2}+2x+1$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;
в-третьих, при $k = 0$ либо ничего не печатается,

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;
в-третьих, при $k = 0$ либо ничего не печатается,

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{3}{0}{}
```
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;
в-третьих, при $k = 0$ либо ничего не печатается,

```
\ComPlusL{0}{2}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{3}{0}{}
```
при компиляции даёт
$2x^2$.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;
в-третьих, при $k = 0$ либо ничего не печатается,
либо печатается 0, если все предыдущие коэффициенты были нулевыми.

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;
в-третьих, при $k = 0$ либо ничего не печатается,
либо печатается 0, если все предыдущие коэффициенты были нулевыми.
`\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{3}{0}{}`
при компиляции даёт

`\ComPlusL{`i`}{`k`}{`e`}` — команда, определенная в `testsample.tex` и `StartTestBigScr.tex` для печати одночлена $k \cdot e$ в составе линейной комбинации аналогичных одночленов.

Значение $i = 3$ отличается от случая $i = 2$ тем, что в случае, когда все остальные слагаемые линейной комбинации имели нулевые коэффициенты, печатается 0 (применяется для последнего слагаемого в сумме):
во-первых, при $k \geqslant 1$ вместо k печатается $+k$;
во-вторых, при $k = -1$ вместо $-1e$ печатается $-e$;
в-третьих, при $k = 0$ либо ничего не печатается,
либо печатается 0, если все предыдущие коэффициенты были нулевыми.
`\ComPlusL{0}{0}{x^{2}}\ComPlusL{1}{0}{x}\ComPlusL{3}{0}{}`
при компиляции даёт
0.

`\ComPlusV{`x`}{`y`}{`z`}` — команда для печати разложения геометрического вектора по базису $\left\{\vec{\mathbf{i}}, \vec{\mathbf{j}}, \vec{\mathbf{k}}\right\}$. Знак первого коэффициента печатается, даже если этот коэффициент положительный.

См.также \ComPlusVb.

Выражение	Результат компиляции
`$\ComPlusV{-1}{2}{3}$`	$-\vec{\mathbf{i}}+2\vec{\mathbf{j}}+3\vec{\mathbf{k}}$
`$\ComPlusV{1}{-2}{3}$`	$+\vec{\mathbf{i}}-2\vec{\mathbf{j}}+3\vec{\mathbf{k}}$
`$\ComPlusV{1}{2}{-3}$`	$+\vec{\mathbf{i}}+2\vec{\mathbf{j}}-3\vec{\mathbf{k}}$
`$\ComPlusV{1}{0}{-1}$`	$+\vec{\mathbf{i}}-\vec{\mathbf{k}}$
`$\ComPlusV{-1}{0}{0}$`	$-\vec{\mathbf{i}}$

`\ComPlusVb{`x`}{`y`}{`z`}` — команда для печати разложения геометрического вектора по базису $\left\{\vec{\mathbf{i}}, \vec{\mathbf{j}}, \vec{\mathbf{k}}\right\}$. Если первый коэффициент положителен, то его знак не печатается.

См.также `\ComPlusV`.

Выражение	Результат компиляции
`$\ComPlusVb{-1}{2}{3}$`	$-\vec{\mathbf{i}}+2\vec{\mathbf{j}}+3\vec{\mathbf{k}}$
`$\ComPlusVb{1}{-2}{3}$`	$\vec{\mathbf{i}}-2\vec{\mathbf{j}}+3\vec{\mathbf{k}}$
`$\ComPlusVb{1}{2}{-3}$`	$\vec{\mathbf{i}}+2\vec{\mathbf{j}}-3\vec{\mathbf{k}}$
`$\ComPlusVb{1}{0}{-1}$`	$\vec{\mathbf{i}}-\vec{\mathbf{k}}$
`$\ComPlusVb{-1}{0}{0}$`	$-\vec{\mathbf{i}}$

`\ComPlusL{`k`}{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n.
Обычно первое слагаемое печатается при $k = 0$.
Компиляция кода

```
\ComPlusL{0}{1}{t}\ComPlusL{0}{2}{t}
\ComPlusL{0}{-1}{t}\ComPlusL{0}{-2}{t}
```

приводит к такому результату: $tt2t-t-2t$

`\ComPlusL{`k`}{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n.

Второе слагаемое при непустом σ обычно печатается при $k = 1$.

Компиляция кода

```
\ComPlusL{0}{1}{t}\ComPlusL{1}{1}{t}\ComPlusL{1}{2}{t}
\ComPlusL{1}{-1}{t}\ComPlusL{1}{-2}{t}
```

приводит к такому результату: $t+t+2t-t-2t$

Результат компиляции кода

```
\ComPlusL{1}{1}{}\ComPlusL{1}{1}{}\ComPlusL{1}{2}{}
\ComPlusL{1}{-1}{}\ComPlusL{1}{-2}{}:
```

$+++2--2$

Результат компиляции кода

```
\ComPlusL{1}{-1}{}\ComPlusL{1}{1}{}\ComPlusL{1}{2}{}
\ComPlusL{1}{-1}{}\ComPlusL{1}{-2}{}:
```

$-++2--2$

`\ComPlusL{`k`}{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n.

Второе слагаемое при пустом σ обычно печатается при $k = 2$ или 3.

Компиляция кода

```
\ComPlusL{0}{1}{t}\ComPlusL{2}{1}{}\ComPlusL{2}{2}{}
\ComPlusL{2}{-1}{}\ComPlusL{2}{-2}{}
```

приводит к такому результату: $t+1+2-1-2$

Результат компиляции кода

```
\ComPlusL{2}{1}{}\ComPlusL{2}{1}{}\ComPlusL{2}{2}{}
\ComPlusL{2}{-1}{}\ComPlusL{2}{-2}{}:
```

$+1+1+2-1-2$

Результат компиляции кода

```
\ComPlusL{2}{-1}{}\ComPlusL{2}{1}{}\ComPlusL{2}{2}{}
\ComPlusL{2}{-1}{}\ComPlusL{2}{-2}{}:
```

$-1+1+2-1-2$

`\ComPlusL{`k`}{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n.

При $k = 4$ эффект чуть отличается: не печатается «+».

Компиляция кода

```
\ComPlusL{0}{1}{t}\ComPlusL{4}{1}{}\ComPlusL{4}{2}{}
\ComPlusL{4}{-1}{}\ComPlusL{4}{-2}{}
```

приводит к такому результату: $t12-1-2$

Результат компиляции кода

```
\ComPlusL{4}{1}{}\ComPlusL{4}{1}{}\ComPlusL{4}{2}{}
\ComPlusL{4}{-1}{}\ComPlusL{4}{-2}{}:
```

$112-1-2$

Результат компиляции кода

```
\ComPlusL{4}{-1}{}\ComPlusL{4}{1}{}\ComPlusL{4}{2}{}
\ComPlusL{4}{-1}{}\ComPlusL{4}{-2}{}:
```

$-112-1-2$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для `$\setcounter{countSign}{0}\ComPlusK{ 2}{x^{2}}`
`\ComPlusK{ 3}{\cos{x}}; (countSign=\arabic{countSign})$`
получаем результат компиляции:
$2x^2+3\cos x; (\texttt{countSign} = 1)$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для `$\setcounter{countSign}{0}\ComPlusK{ 1}{x^{2}}`
`\ComPlusK{ 3}{\cos{x}}; (countSign=\arabic{countSign})$`
получаем результат компиляции:
$1x^2+3\cos x; (\texttt{countSign} = 1)$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для `$\setcounter{countSign}{0}\ComPlusK{-1}{x^{2}}`
`\ComPlusK{ 3}{\cos{x}}; (countSign=\arabic{countSign})$`
получаем результат компиляции:
$-1x^2+3\cos x; (\texttt{countSign} = 1)$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для
```
$\setcounter{countSign}{0}\ComPlusK{-2}{x^{2}}
\ComPlusK{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:

$-2x^2+3\cos x; (\texttt{countSign} = 1)$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для
```
$\setcounter{countSign}{0}\ComPlusK{ 0}{x^{2}}
\ComPlusK{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:

$3\cos x; (\texttt{countSign} = 1)$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для
```
$\setcounter{countSign}{0}\ComPlusK{ 1}{x^{2}}
\ComPlusK{-3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:

$1x^2-3\cos x; (\texttt{countSign} = 1)$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для
```
$\setcounter{countSign}{0}\ComPlusK{ 0}{x^{2}}
\ComPlusK{-3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:

$-3\cos x; (\texttt{countSign} = 1)$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для
```
$\setcounter{countSign}{0}\ComPlusK{ 0}{x^{2}}
\ComPlusK{ 0}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:

$\setcounter{\texttt{countSign}}{0}\ComPlusK{ 0}{x^{2}}\ComPlusK{ 0}{\cos{x}}; (\texttt{countSign}=\arabic{\texttt{countSign}})$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для `$\setcounter{countSign}{0}\ComPlusK{-1}{x^{2}}`
`\ComPlusK{-3}{\cos{x}}; (countSign=\arabic{countSign})$`
получаем результат компиляции:
$-1x^2-3\cos x; (\texttt{countSign} = 1)$

`\ComPlusK{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`

$$\begin{cases} \texttt{CountSign} = 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 0, \\ \text{ни } n = 0, \text{ ни } \sigma\text{не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma, \text{ причём } + \text{ не печатается.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} = 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n = 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{ни } 0, \text{ ни } \sigma \text{ не печатаются.} \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n > 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } + n\sigma. \end{cases}$$

$$\begin{cases} \texttt{CountSign} \neq 0, \\ n < 0 \end{cases} \Rightarrow \begin{cases} \texttt{CountSign} = 1, \\ \text{печатается } n\sigma. \end{cases}$$

Например, для
```
$\setcounter{countSign}{0}\ComPlusK{-1}{x^{2}}
\ComPlusK{ 0}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:

$-1x^2; (\texttt{countSign} = 1)$

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **`countSign`** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{ 2}{x^{2}}
\ComPlusN{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$2x^2+3\cos x;$ (**`countSign`** $= 1$)

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **`countSign`** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{ 1}{x^{2}}
\ComPlusN{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$x^2+3\cos x; (\mathbf{countSign} = 1)$

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для `countSign` $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{-1}{x^{2}}
\ComPlusN{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-x^2+3\cos x;$ (`countSign` $= 1$)

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **countSign** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{-2}{x^{2}}
\ComPlusN{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-2x^2+3\cos x; (\textbf{countSign} = 1)$

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **`countSign`** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{ 0}{x^{2}}
\ComPlusN{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$3\cos x; (\mathbf{countSign} = 1)$

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **`countSign`** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{ 1}{x^{2}}
\ComPlusN{-3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$x^2-3\cos x;$ (**`countSign`** $= 1$)

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **`countSign`** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{ 0}{x^{2}}
\ComPlusN{-3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-3\cos x; (\mathbf{countSign} = 1)$

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **`countSign`** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{ 0}{x^{2}}
\ComPlusN{ 0}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$\setcounter{countSign}{0}\ComPlusN{ 0}{x^{2}}\ComPlusN{ 0}{\cos{x}}; (\mathbf{countSign} = \arabic{countSign})$

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **countSign** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{-1}{x^{2}}
\ComPlusN{-3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-x^2-3\cos x;$ (**countSign** $= 1$)

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **countSign** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{-1}{x^{2}}
\ComPlusN{ 0}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-x^2; (\textbf{countSign} = 1)$

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **countSign** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{-1}{x^{2}}
\ComPlusN{ 1}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-x^2+\cos x;$ (**countSign** $= 1$)

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **countSign** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{-1}{x^{2}}
\ComPlusN{-1}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-x^2 - \cos x; (\textbf{countSign} = 1)$

`\ComPlusN{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusK`, кроме того, что для **`countSign`** $= 0$
при $n = 1$ печатается σ вместо 1σ,
$+\sigma$ вместо $+1\sigma$,
при $n = -1$ печатается $-\sigma$ вместо -1σ.
Например, для
```
$\setcounter{countSign}{0}\ComPlusN{ 1}{x^{2}}
\ComPlusN{ 1}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$x^2 + \cos x; (\mathbf{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{ 2}{x^{2}}
\ComPlusM{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$2x^2 3\cos x; (\texttt{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для `$\setcounter{countSign}{0}\ComPlusM{ 1}{x^{2}}`
`\ComPlusM{ 3}{\cos{x}}; (countSign=\arabic{countSign})$`
получаем результат компиляции:
$1x^{2}3\cos x; (\mathtt{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{-1}{x^{2}}
\ComPlusM{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-1x^{2}3\cos x; (\texttt{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{-2}{x^{2}}
\ComPlusM{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-2x^{2}3\cos x; (\mathtt{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{ 0}{x^{2}}
\ComPlusM{ 3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$3\cos x; (\texttt{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{ 1}{x^{2}}
\ComPlusM{-3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$1x^{2}-3\cos x;(\mathbf{countSign}=1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{ 0}{x^{2}}
\ComPlusM{-3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-3\cos x; (\mathbf{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для `$\setcounter{countSign}{0}\ComPlusM{ 0}{x^{2}}`
`\ComPlusM{ 0}{\cos{x}}; (countSign=\arabic{countSign})$`
получаем результат компиляции:
; (**countSign** $= 0$)

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{-1}{x^{2}}
\ComPlusM{-3}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-1x^2-3\cos x; (\mathbf{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{-1}{x^{2}}
\ComPlusM{ 0}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-1x^2; (\texttt{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{-1}{x^{2}}
\ComPlusM{ 1}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-1x^{2}1\cos x; (\mathtt{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{-1}{x^{2}}
\ComPlusM{-1}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$-1x^2-1\cos x; (\mathbf{countSign} = 1)$

`\ComPlusM{`n`}{`σ`}` — команда для печати выражения σ с коэффициентом n. Результат зависит от значения **счётчика** `countSign`
Результат определяется приктически теми же правилами, что и `\ComPlusM`, кроме того, что знак $+$ перед положительным n не печатается ни при каких значениях **счётчика** `countSign`.
Например, для
```
$\setcounter{countSign}{0}\ComPlusM{ 1}{x^{2}}
\ComPlusM{ 1}{\cos{x}}; (countSign=\arabic{countSign})$
```
получаем результат компиляции:
$1x^2 1\cos x; (\texttt{countSign} = 1)$

`\ComRealPlus{`a`}{`b`}{`c`}{`d`}{nameA}{nameB}` — команда для вычисления выражения
$a \cdot 10^b + \left(c \cdot 10^d\right) = \underbrace{\texttt{\textbackslash nameA}}_{\text{4 знака}} \cdot 10^{\texttt{\textbackslash nameB}}$.

Эта команда позволяет работать с действительными числами в «научной форме» с четырьмя значащими цифрами.

Например, в результате компиляции получим:

`\ComRealPlus({1}{2}{3}{4}{Abc}{Xx})`
`1\cdot10^{2}+3\cdot10^{4}=\Abc\cdot10^{\Xx}`
получим $1 \cdot 10^2 + 3 \cdot 10^4 = 301 \cdot 10^2$.

`\ComRealPlus({6}{4}{3}{2}{Uvw}{Yy})`
`6\cdot10^{4}+3\cdot10^{2}=\Abc\cdot10^{\Xx}`
получим $6 \cdot 10^4 + 3 \cdot 10^2 = 603 \cdot 10^2$.

`\ComRealPlus({9}{6}{3}{3}{Pqr}{Zz})`
`9\cdot10^{6}+3\cdot10^{3}=\Abc\cdot10^{\Xx}`
получим $9 \cdot 10^6 + 3 \cdot 10^3 = 9003 \cdot 10^3$.

`\ComRealMult{`a`}{`b`}{`c`}{`d`}{nameA}{nameB}` — команда для вычисления выражения
$a \cdot 10^b \cdot \left(c \cdot 10^d\right) = \underbrace{\backslash\texttt{nameA}}_{\text{4 знака}} \cdot 10^{\backslash\texttt{nameB}}$.

Эта команда позволяет работать с действительными числами в «научной форме» с четырьмя значащими цифрами.

Например, в результате компиляции получим:

```
\ComRealMult({1}{2}{3}{4}{Abc}{Xx})
1\cdot10^{2}\cdot3\cdot10^{4}=\Abc\cdot10^{\Xx}
```

получим $1 \cdot 10^2 \cdot 3 \cdot 10^4 = 3 \cdot 10^6$.

```
\ComRealMult({6}{4}{3}{2}{Uvw}{Yy})
6\cdot10^{4}\cdot3\cdot10^{2}=\Abc\cdot10^{\Xx}
```

получим $6 \cdot 10^4 \cdot 3 \cdot 10^2 = 18 \cdot 10^6$.

```
\ComRealMult({9}{6}{3}{3}{Pqr}{Zz})
9\cdot10^{6}\cdot3\cdot10^{3}=\Abc\cdot10^{\Xx}
```

получим $9 \cdot 10^6 \cdot 3 \cdot 10^3 = 27 \cdot 10^9$.

`\ComRealDivi{`a`}{`b`}{`c`}{`d`}{nameA}{nameB}` — команда для вычисления выражения $a \cdot 10^b : \left(c \cdot 10^d\right) = \underbrace{\texttt{\textbackslash nameA}}_{\text{4 знака}} \cdot 10^{\texttt{\textbackslash nameB}}$.

Эта команда позволяет работать с действительными числами в «научной форме» с четырьмя значащими цифрами, причем производится не округление, а отсечение «лишних цифр».

```
\ComRealDivi({1}{2}{3}{4}{Abc}{Xx})
\displaystyle\frac{1\cdot10^{2}}{3\cdot10^{4}} =\Abc\cdot10^{\Xx}.
```

Результат компиляции: $\frac{1 \cdot 10^2}{3 \cdot 10^4} = 3333 \cdot 10^{-6}$.

```
\ComRealDivi({6}{4}{3}{2}{Uvw}{Yy})
\displaystyle\frac{6\cdot10^{4}}{3\cdot10^{2}} =\Abc\cdot10^{\Xx}.
```

Результат компиляции: $\frac{6 \cdot 10^4}{3 \cdot 10^2} = 2000 \cdot 10^{-1}$.

```
\ComRealDivi({9}{6}{3}{3}{Pqr}{Zz})
\displaystyle\frac{9\cdot10^{6}}{3\cdot10^{3}} =\Abc\cdot10^{\Xx}.
```

Результат компиляции: $\frac{9 \cdot 10^6}{3 \cdot 10^3} = 30000 \cdot 10^0$.

\GenerateNonZeroParamsZ{a_1}{a_2}{a_3}{a_4}{a_5}{a_6}{a_7}{a_8}{a_9} — команда для генерирования ненулевых значений регистров
\CountVax, \CountVay, \CountVaz,
\CountVbx, \CountVby, \CountVbz,
\CountVcx, \CountVcy, \CountVcz,
\CountVvx, \CountVvy, \CountVvz.
Эти значения определяются с помощью параметров $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9$. Параметры могут задаваться и как содержимое **счетчика** или **регистра**, кроме \the\CountSca и \the\CountTmp.
Аргументы этой команды могут задаваться и как содержимое **счетчика** или **регистра**, кроме \the\CountSca и \the\CountTmp.

`\ComNumbOfPlaceA{`m`}{`n`}` — команда для вычисления числа сочетаний $A_n^m = (m+1)(m+2)\dots(n-1)n = \dfrac{n!}{m!}$.
Результат присваивается счётчику `\CountSca`.
Аргументы этой команды могут задаваться и как содержимое **счетчика** или **регистра**, кроме `\the\CountSca` и `\the\CountTmp`.

`\ComScalProd{`u_1`}{`u_2`}{`u_3`}{`v_1`}{`v_2`}{`v_3`}` — команда для вычисления скалярного произведения векторов $u_1\vec{\mathbf{i}}+u_2\vec{\mathbf{j}}+u_3\vec{\mathbf{k}}$ и $v_1\vec{\mathbf{i}}+v_2\vec{\mathbf{j}}+v_3\vec{\mathbf{k}}$. Результат присваивается **регистру** `\CountSca`, т.е.
$\left((\#1)\vec{\mathbf{i}}+(\#2)\vec{\mathbf{j}}+(\#3)\vec{\mathbf{k}},\ (\#4)\vec{\mathbf{i}}+(\#5)\vec{\mathbf{j}}+(\#6)\vec{\mathbf{k}}\right)=$ `\the\CountSca`.
Аргументы этой команды могут задаваться и как содержимое **счетчика** или **регистра**, кроме `\the\CountSca` и `\the\CountTmp`.

`\ComVectProd{`u_1`}{`u_2`}{`u_3`}{`v_1`}{`v_2`}{`v_3`}` — команда для вычисления векторного произведения векторов $u_1\vec{\mathbf{i}}+u_2\vec{\mathbf{j}}+u_3\vec{\mathbf{k}}$ и $v_1\vec{\mathbf{i}}+v_2\vec{\mathbf{j}}+v_3\vec{\mathbf{k}}$. Результат присваивается **регистрам** `\CountVvx`, `\CountVvy`, `\CountVvz`, т.е.

$\left[(\#1)\vec{\mathbf{i}}+(\#2)\vec{\mathbf{j}}+(\#3)\vec{\mathbf{k}},\ (\#4)\vec{\mathbf{i}}+(\#5)\vec{\mathbf{j}}+(\#6)\vec{\mathbf{k}}\right]=$

$=$ `\the\CountVvx` $\vec{\mathbf{i}}+$ `\the\CountVvy` $\vec{\mathbf{j}}+$ `\the\CountVvz` $\vec{\mathbf{k}}$.

Аргументы этой команды могут задаваться и как содержимое **счетчика** или **регистра**, кроме `\the\CountSca` и `\the\CountTmp`.

`\DivideRealA{`a`}{`b`}{`c`}{`d`}{`$ComA$`}{`$ComB$`}{`e`}` — команда для вычисления частного десятичных дробей:
$\frac{a\cdot 10^{-b}}{c\cdot 10^{-d}} \approx$ `\ComA` $\cdot 10^{-$`\ComB`$}$, где `\ComB` $= e$ и последние e цифр числа `\ComA` образуют округленную до e цифр мантиссу числа $\frac{a\cdot 10^{-b}}{c\cdot 10^{-d}}$. Команды `\ComA`, `\ComB` определяются при выполнении этой команды.

Например, компиляция кода

```
\DivideRealA{12345}{2}{6789}{4}{Abc}{Pqr}{3}
```

приводит к появлению команды `\Abc`, определенной как число 181838, и команды `\Pqr`, определенной как число 3, причём
$\frac{12345\cdot 10^{-2}}{6789\cdot 10^{-4}} = \frac{123.45}{0.6789} \approx 181.838 = 181838\cdot 10^{-3}$.

В качестве аргументов команды нельзя использовать `\the\CountTmp`, `\the\CountSca` и `\arabic{countSign}`.

\MultiRealA{a}{b}{c}{d}{$ComA$}{$ComB$}{e} — команда для вычисления произведения десятичных дробей:
$\left(a\cdot 10^{-b}\right)\cdot\left(c\cdot 10^{-d}\right)\approx \backslash\texttt{ComA}\cdot 10^{-\backslash\texttt{ComB}}$, где $\backslash\texttt{ComB}=e$ и последние e цифр числа \ComA образуют округленную до e цифр мантиссу числа $\left(a\cdot 10^{-b}\right)\cdot\left(c\cdot 10^{-d}\right)$.
Команды \ComA, \ComB определяются при выполнении этой команды.

Например, компиляция кода

```
\MiltiRealA{12345}{2}{6789}{4}{Lk}{Nm}{3}
```

приводит к появлению команды \Lk, определенной как число 83810, и команды \Nm, определенной как число 3, причём
$12345\cdot 10^{-2}\cdot 6789\cdot 10^{-4}=123.45\cdot 0.6789\approx 83.810=83810\cdot 10^{-3}$.

В качестве аргументов команды нельзя использовать \the\CountTmp, \the\CountSca и \arabic{countSign}.

`\MakeParamAnysign{`a_1`}{`a_2`}{`a_3`}{`a_4`}{`a_5`}{`a_6`}{`a_7`}{`p`}{`q`}` — команда для генерирования значения натурального числа из диапазона $\{-q;\ -q+1; \ldots; -p-1\} \cup \{p+1;\ p+2; \ldots; q\}$.

Результат присваивается **регистру** `\CountTmp`. Параметры могут задаваться и как содержимое **счетчика** или **регистра**, кроме `\the\CountSca` и `\the\CountTmp`.

`\MakeParamPositiv{`a_1`}{`a_2`}{`a_3`}{`a_4`}{`a_5`}{`a_6`}{`a_7`}{`p`}{`q`}` — команда для генерирования значения натурального числа из диапазона $\{p;\ p+1;\ldots;q\}$.

Результат присваивается **регистру** `\CountTmp`. Параметры могут задаваться и как содержимое **счетчика** или **регистра**, кроме `\the\CountSca` и `\the\CountTmp`.

`\NOD{`m`}{`n`}` — команда для вычисления наибольшего общего делителя чисел m и n. Результат присваивается **регистру** `\CountSca`. m и n могут задаваться и как содержимое **счетчика** или **регистра**, кроме `\the\CountSca` и `\the\CountTmp`.

`\NOD{6}{9}`, `\the\CountSca` $= 3$.
`\NOD{6}{-9}`, `\the\CountSca` $= 3$.
`\NOD{-6}{9}`, `\the\CountSca` $= 3$.
`\NOD{-6}{-9}`, `\the\CountSca` $= 3$.
`\NOD{6}{8}`, `\the\CountSca` $= 2$.
`\NOD{6}{-8}`, `\the\CountSca` $= 2$.
`\NOD{-6}{8}`, `\the\CountSca` $= 2$.
`\NOD{-6}{-8}`, `\the\CountSca` $= 2$.

`\PrintRealA{`n`}{`k`}` — команда для печати вместо целого числа n десятичной дроби, в котором последние k цифр числа n представлены в качестве как мантиссы, т.е. печатается в виде десятичной дроби число $\frac{n}{10^k}$. В качестве разделителя целой части и мантиссы, в отличие от `\PrintRealB`, используется точка.

В качестве n и k нельзя использовать `\the\CountTmp` и `\the\CountSca`.

Например, компиляция кода

```
\PrintRealA{123456789}{0}; \PrintRealA{123456789}{1};
\PrintRealA{123456789}{2};\PrintRealA{123456789}{3};
```

приводит к результату: 123456789.0; 12345678.9; 1234567.89; 123456.789

Команду `\PrintRealA` нельзя использовать в составе команд `\RespBoxMath` и т.п., поскольку результат выполнения команды `\PrintRealA` не воспринимается TEX-ом как число или формула.

`\PrintRealB{`n`}{`k`}` — команда для печати вместо целого числа n десятичной дроби, в котором последние k цифр числа n представлены в качестве как мантиссы, т.е. печатается в виде десятичной дроби число $\frac{n}{10^k}$. В качестве разделителя целой части и мантиссы, в отличие от `\PrintRealA`, используется запятая.

В качестве n и k нельзя использовать `\the\CountTmp` и `\the\CountSca`.

Например, компиляция кода

```
\PrintRealB{123456789}{0}; \PrintRealB{123456789}{1};
\PrintRealB{123456789}{2};\PrintRealB{123456789}{3};
```

приводит к результату: 123456789,0; 12345678,9; 1234567,89; 123456,789

Команду `\PrintRealB` нельзя использовать в составе команд `\RespBoxMath` и т.п., поскольку результат выполнения команды `\PrintRealB` не воспринимается TEX-ом как число или формула.

`\SummaRealA{`a`}{`b`}{`c`}{`d`}{`$ComA$`}{`$ComB$`}` — команда для вычисления суммы десятичных дробей:
$a \cdot 10^{-b} + c \cdot 10^{-d} = \texttt{\ComA} \cdot 10^{-\texttt{\ComB}}$.
Команды `\ComA`, `\ComB` определяются при выполнении этой команды.

Например, компиляция кода

```
\SummaRealA{12345}{2}{6789}{3}{Aa}{Bb}
```

приводит к появлению команды `\Aa`, определенной как число 130239, и команды `\Bb`, определенной как число 3, причём
$12345 \cdot 10^{-2} + 6789 \cdot 10^{-3} = 123.45 + 6.789 = 130.239 = 130239 \cdot 10^{-3}$.

В качестве аргументов команды нельзя использовать `\the\CountTmp`, `\the\CountSca` и `\arabic{countSign`.

V. Некоторые команды пакета acrotex

`\Ans` — «открывающая скобка» для варианта ответа в окружении `manswers`.

`\bChoices` — «открывающая скобка» для блока вариантов ответов в окружении `manswers`.

`\eAns` — «закрывающая скобка» для варианта ответа в окружении `manswers`.

`\eChoices` — «закрывающая скобка» для блока вариантов ответов в окружении `manswers`.

`manswers` — окружение для формирования теста с множественными вариантами верных ответов. В результате компиляции кода:

```
\begin{manswers}{3}
Отметить щелчком левой кнопки мыши в «квадратике» все те выражения,
которые эквивалентны выражению \mbox{$x^2-2x$}:
  \bChoices[nCols=2]
    \Ans1 $x(x-2)$ \eAns
    \Ans0 $(x-2)^{2}$ \eAns
    \Ans0 $(x+1)^{2}-1$\eAns
    \Ans0 $(x-1)^{2}+1$\eAns
    \Ans1 $(x-1)^{2}-1$\eAns
    \Ans0 $(x-1)^{2}$ \eAns
  \eChoices
\end{manswers}
```

получим тест

Отметить щелчком левой кнопки мыши в «квадратике» все те выражения, которые эквивалентны выражению $x^2 - 2x$*:*

☐ $x(x-2)$ ☐ $(x-2)^2$ ☐ $(x+1)^2-1$

☐ $(x-1)^2+1$ ☐ $(x-1)^2-1$ ☐ $(x-1)^2$

`\coeff{`n`}` — команда для формирования поля для ввода в окружении `mathGrp`.

Здесь n — натуральное число, равное значению, ввод которого в данное поле засчитывается как верный ответ.

`mathGrp` — название окружения, введенного в пакете `acrotex`, запускаемого при использовании
`\input testsample.tex`.

```
  \begin{mathGrp}\PTs*{m}
     ...\coeff{...}...
\end{mathGrp}
```

Здесь m — число баллов, начисляемое за каждое правильно заполненное поле для ввода, определяемое командой `\coeff`.

Таким образом, количество баллов, начисляемых за полное решение задания, равно произведению m на число применений команды `\coeff` внутри окружения `mathGrp`.

`nCols` — параметр в окружении `manswers` задающий число столбцов для определения числа столбцов в вариантах ответов.

`\RespBoxMath{`F`}(`V`){`n`}{`ε`}{`Set`}` — команда для создания поля, предназначенного для ввода математического выражения.

Например, компиляция фрагмента

```
$\varphi(p,q)=\RespBoxMath{p^2*sqrt(q)}(pq){3}{.000001}{[2,3]x[-1,1]}$
```

приведёт к результату вида

$\varphi(p,q) =$ [ ],

где введенное в поле выражение будет сравниваться с выражением $p^2 \cdot \sqrt{q}$ с помощью сравнения с точностью до $0,000001$ значений эталонного и введенного тестируемым выражений в точках

$(p,q) \in \{2;\ 2,5;\ 3\} \times \{-1;\ 0;\ 1\} =$

$= \{(2;-1), (2;0), (2;1), (2,5;-1), (2,5;0), (2,5;1), (3;-1), (3;0), (3;1)\}.$

Предметный указатель

Предметный указатель

Спасибо

за

внимание!

e-mail: melnikov@k66.ru, melnikov@r66.ru

сайты: http://melnikov.k66.ru, http://melnikov.web.ur.ru

Перейдем: к **описанию TEXнологии** `OOPattern`,

к **примерам по TEXнологии** `PrimPattern`?