Вступ до видавничої системи LaTeX

Власенко Д.І., Курінний Г.Ч.

Зміст

Частина I
З чого починається LaTeX 2ε

Коли в руках молоток, все бачиться як цвях.

Наш курс присвячений ознайомленню з видавничою системою LaTeX 2_ε, що створена на базі TeX'у. Основне призначення системи LaTeX 2_ε — забезпечення потреб наукового листування, підготовка рукописів наукових статей, видавництво наукових журналів, наукових та навчальних книг та посібників.

Побрібно знати відміни між текстовими редакторами та видавничими системами.

Текстовий редактор — це програма, яка дозволяє вводити в комп'ютер текст для його подальшого зберігання, друку та передачі на інший комп'ютер. Навіть найпростіші текстові редактори мають додаткові можливості по оформленню текста (вирівнювання, табуляція, тощо), малювання найпростіших зображень та створення найпростіших таблиць. Те, що ви бачите на екрані після набирання тексту в тектовому редакторі, ви побачите і після друку. Така властивість програми (друкується те, що видно на екрані) називається «wysiwyg»[1]. Потужні текстові редактори (тобто текстові редактори з багатьма додатковими можливостями) за своїми можливостями наближаються до видавничих систем.

Видавнича система (або система верстки) — це пакет програм, що призначені для оформлення уже створених елементів публікації (оформлення тексту; вставка таблиць, формул, графіки; створення розділів, підрозділів, предметних показчиків, та інше). У всі виданичі системи входить редактор тексту. Такий редактор звичайно слабкий, з малими власними можливостями. Вважається, що всі елементи публікації можуть бути створені поза межами видавничої системи — тією програмою чи пакетом, що найзручніша для відповідного елемента.

[1] What you see is what you get.

Почнемо з того, що обговоримо можливості системи вестки, які потрібно враховувати при її виборі. Після цього повернемо погляди на нашого героя — LaTeX 2$_\varepsilon$, і прослідкуємо його епічний шлях. Завершимо ж вступ простим прикладом. Ну, що? Пораа$_{a_{a_{a_{a_{a}}}}}$

1 Видавничі системи. Вимоги до них

1. **Адекватне представлення інформації**

   Видавнича система повинна дозволяти користувачеві здійснити будь-яку забаганку, що стосується оформлення друкованого слова. Степінь легкості, з якою ви можете здійснити задумане в різних видавничах системах буде різною. Для прикладу, у видавничій системі PageMaker вам буде коштувати значно більших зусиль вставити в текст символ ∞ ніж у LaTeX 2$_\varepsilon$, а в TeX-документі змінити шрифт і розмір букв окремого слова буде важче, ніж в текстовому редакторі Word.

2. **Автоматизація однотипної роботи**

   Всі видавничі системи мають тією чи іншою мірою доступні для малодосвідченого користувача можливості автоматичної нумерації структурних елементів (частин, розділів, підрозділів, додатків, формул, ілюстрацій, таблиць); можливості зручного створення посилань (на формулу, на сторінку, на розділ, на літературне джерело); можливості створення переліку цитованих джерел, переліку ілюстрацій, змісту, предметного показчика.

3. **Можливості швидкого переформатування документа**

   Якщо шаблон чи стиль документа не задовольняє вимог користувача, то зусилля для зміни шаблону або стилю повинні бути якнайменші.

4. **Переносимість створеного документу**

   Створений документ після друку повинен мати один і той же вигляд не-

залежно від використаної операційної системи та обладнання. Потрібно давати собі звіт щодо можливості редагування чи то окремих елементів документа, чи всього документа на комп'ютері з іншим програмним забезпеченням, з іншою операційною системою.

5. **Вартість необхідного програмного забезпечення**

   Чим менше, тим краще. Варто також звернути увагу на те, що основне програмне забезпечення супроводжується додатковим (яке має свою вартість) і, крім того, програмне забезпечення може працювати лише на обчислювальних машинах певного типу (на певній платформі).

Варто також звернути увагу

- чи використовується система WYSIWYG.
- наскільки часто приходиться використовувати мишку при роботі з документом.

2 Дещиця про видавничу справу та про створення TEX

Для того, щоб надрукуватися, автори віддають свої рукописи у видавництво. Потім дизайнер видавництва визначає макет документа (ширину смуги набору, шрифти, інтервали над і під заголовками та інше). Дизайнер записує свої вимоги в рукопис і віддає рукопис верстальщику, який верстає книгу у відповідності до цих вимог.

Дизайнер-людина намагається зрозуміти задум автора, коли той писав рукопис. Він визначає форматування заголовків розділів, цитат, прикладів, формул та іншого виходячи із свого професійного досвіду та із змісту рукопису.

Ланцюжок «автор-дизайнер-верстальщик» повинен правильно грати в «зрозумій мене». Доналд Кнут (Donald E. Knuth) почав працювати над видавничою системою TEX у травні 1977 року через відверте незадоволення

тим, що Американське Математичне Товариство ($\mathcal{AMS}$) робило з його статтями в процесі їх публікації. Десь в 1974 році він навіть припинив надсилати статті: «просто мені було надто боляче дивитися на кінцевий результат». Фактично Кнут створив спеціалізовану мову програмування, на основі якої створюються відавничі системи. Доналд Кнут переконливо рекомендував українською мовою читати послідовність латинських букв TEX українською мовою як тех, а не текс.

На початку вісімдесятих років Лесли Лампорт (Leslie Lamport) почав роботу над видавничою системою LATEX, в основу якої було покладено TEX. Ця система дозволяє користувачеві зосередитися на структурі документа, а не на його форматуванні. Для набору складних математичних текстів Майклом Співаком (M. Spivak) для Американського Математичного Товариства було розроблено пакет $\mathcal{AMS}$-LATEX, який створювався на основі TEX. Але, на відміну від LATEX, цей пакет не має сильних засобів управління версткою текста. На сьогодні існує ще ряд пакетів на основі TEX, які використовуються для верстки. Зауважимо, що TEX, створений Кнутом ще називають Plain TEX'ом.

1989 року команда творців LATEX3 на чолі з Франком Мітельбахом (Frank Mittelbach) почала працювати над об'єднанням різних версій та розширенням можливостей LATEX'а. Проміжний варіант їхньої работи — версія LATEX 2_ε випущено в 1994 році. В ній були зроблені деякі давно очікувані спрощення і об'єднанні всі варіанти LATEXа, що розійшлися після того, як кілька років тому була випущена версія LaTeX 2.09. Щоб не плутати цю нову версію зі старою, вона одержала назву LATEX 2_ε. Ми будемо обговорювати роботу саме з LATEX 2_ε.

Оскільки українською мовою TEX потрібно читати як тех, то і LATEX 2_ε читається як Латех-епсілон.

3 Переваги та вади LaTeX'а

Недоліки є продовженням достоїнств.

Ми з'ясували, що TeX є мова програмування, тому природно, що документ створений TeX є звичайний текстовий файл, що не містить ніяких прихованих керуючих символів чи команд. Тому TeX-документи вигідно відрізняються переносимістю та платформною незалежністю. Переносимість полягає в тому, що TeX-документ буде мати однаковий вигляд на якому б друкарському пристрої він не був би надрукований. TeX-системи створені для різних операційних систем — DOS, Windows, Linux, Unix, Mac, OS/2. До того ж, створені документи можуть бути програмно переведені в PostScript- або в PDF-документ. Додамо, що конвертор (програма-перетворювач) в PDF, як і власне TeX-системи, безкоштовний.

Для створення домументів в LaTeX 2_ε передбачено набір стандартних шаблонів — лист, стаття, книга. Ви можете створити свій власний шаблон для задоволення ваших конкретних потреб. Наприклад, це може бути шаблон для дисертації, що відповідає вимогам ВАК'а[2].

Звичайно, у різних математичних, фізичних та технічних журналах свої власні вимоги до оформлення статей. Такі журнали розробили відповідні шаблони (стильові файли), які приймають участь у створені кінцевого результату — правильно оформленої статті.

Можливе внесення змін в уже існуючі шаблони. Правильний вибір шаблону дозволяє зосередитися на змістовній частині документа та на його логічній структурі (підпорядкування частин), а не відвертати увагу на художню діяльність з оформлення документу — ця задача лежить на плечах TeXа. Із особливим смаком TeX створює математичні формули — його цьому спеціально вчили.

Впевнено можна сказати, що TeX є дуже TeXнічним помічником, що го-

[2]Вища атестаційна комісія. Звичайно, дисертації, що подаються у ВАК, повинні бути оформленими за правилами ВАК'а.

товий забрати у Вас всю TEXнічну роботу. Але, щоб наші твердження не були просто словами, доречно поставити дослід. Наберемо в системі LaTeX 2_ε рядок

$$o_1, o^1, o_1^2, \overset{*}{o}, \underset{*}{o}, \underset{*}{\overset{*}{o}}, o^j{}_i, \dddot{o}_1^2, \ddot{o}_1^2, \acute{o}, \acute{\acute{o}}, \bar{\bar{o}}, \breve{\breve{o}}, \check{\check{o}}, \ddot{\ddot{o}}, \tilde{\tilde{o}}, o, o, o,\ o, o,\ o, \quad o, \qquad o, \widehat{oo}, \OE, \c{o}, \u{o}, \u{\c{o}}$$

Для набору такого рядка потрібно в найпростішому текстовому редакторі набрати текст

```
  $$o_1,o^1,o_1^2,\overset{*}{o},\underset{*}{o},
\overset{*}{\underset{*}{o}},o^j_{\quad i},\dddot{o_1^2},   \dddot{o}_1^2,
   \Acute{o},\Acute{\Acute{o}},\Bar{\Bar{o}},\Breve{\Breve{o}},
\Check{\Check{o}},\Ddot{\Ddot{o}},\Tilde{\Tilde{o}},o,\! o,o,\,
 o,\:o,\; o,\quad o, \qquad o, \mbox{\t oo ,\OE, \c{o},\u{o},\u{\c{o}}}
$$
```

що містить біля 300 символів. Відповідний текстовий файл буде містити біля 300 байтів. Отже маємо результат: потрібний рядок професійна друкарка набере за 1 хвилину на будь-якій обчислювальній машині і відповідний файл буде містити (якщо вона не використовує надто потужний редактор) біля 300 байтів. Для перетворення надрукованого рядка в типографський рядок візьмемо безплатно ліцензований пакет LaTeX 2_ε встановимо на будь-якому комп'ютері на безплатну операційну систему і роздрукуємо на будь-якому принтері. Далі попросимо недосвідченого у видавничих справах товариша набрати той же рядок.

Далі купимо сучасний редактор Word, спробуємо за годину набрати рядок у цьому редакторі, оцінимо його величину і спробуємо роздрукувати в операційній ситемі DOS. І, нарешті, недосвідченого у використанні Word товариша попросимо набрати потрібний рядок.

Такий дослід чи явний чи подумки переконає в перевагах LaTeX 2_ε.

Коли математик користується пакетом символьних обчислень для одержання певних формул, а потім результат бажає вставити в статтю чи книгу, то йому не потрібно формули переписувати руками — практично всі пакети символьних обчислень надають можливість зберегти результат у TEX-форматі.

Повагу до вміння LaTeX 2ε верстати формули демонструє і Word — останні версії цього редактора створюють формули за допомогою LaTeX 2ε.

Як вже було зауважено, недоліки LaTeX 2ε є продовженнями його переваг. У практиці видавництва художньої чи популярної літератури не зустрінеться випадок, коли потрібно набирати рядок, що схожий на наведений, а в науковому виданні такі екзотичні позначення є звичайною річчю. Максимальна зручність для наукових публікацій, для створення бездоганно оформлених статей малодосвідченими у видавничих справах науковцями обертається незручністю при використанні у домашньому господарстві чи у створенні художніх альбомів. Подібним чином досконалі годинники значно програють звичайному кусочкові свинцю при використані в якості важка для вудочки.

Все, сказане вище, зробило видавничі системи системи на основі TeXa стандартом de facto для наукових видавництв. Фізико-математичні журнали всього світу приймають для опублікування статті, які підготовані в LaTeX. В той же час видавництво художньої літератури може навіть не знати про існування такої видавничої системи.

4 Рекомендована література

Рекомендуємо російськомовну літературу, оскільки 1) україномовна нам невідома[3] (не виключено, що вона не існує): 2) публікації російською і українською мовами мають суттєву спільну проблему — використання кирилиці: 3) не такі вже ми й великі спеціалісти в інших мовах.

Гуссенс М., Миттельбах Ф., Самарин А. Путеводитель по пакету LaTeX и его расширению LaTeX 2ε. — М., Мир, (1999), 606 стр.

Ця книга — цінний довідниковий посібник від розробників LaTeX3. Розглянуто багато пакетів, що розширюють можливості основного пакета для TeX-спеціалістів, що постійно мають справу з LaTeXом. Текст в книзі, що

[3] В пакет MikTeX входят переклади посібника Tobias Oetiker «Не надто короткий вступ до LaTeX 2ε», в тому чіслі і українською мовою, в перекладі Максима Полякова

стосується пакетів, має багато спільного із стандартною документацією до LaTeX 2$_\varepsilon$. Природно, документацію можна знайти[4] на CTAN'і. Але знайомство з LaTeXом краще починати з двох наступних книг:

Котельников И. А., Чеботаев П. З. LaTeX 2$_\varepsilon$ по-русски. — Новосибирск, Новосибирский Хронограф, (2004), 496 стр.

Львовский С. М. Набор и верстка в пакете LaTeX, 2-е издание. — М., Космосинформ, (1995), 374 стр.

І ще дві книги, які вже можна віднести до класики:

Спивак М. Восхитительный TeX: руководство по комфортному изготовлению научных публикаций в пакете $\mathcal{AMS}$-TeX. — М., Мир, (1993).

Кнут Д. Е. Все про TeX. — Протвино, РДТеХ, (1993).

5 Ресурси LaTeX в мережі INTERNET

Вихідний сайт користувачів TeXa (TeX Users Group) — http://www.tug.org. Тут, так чи інакше, можна знайти практично все, що відноситься до TeX. Існує міжнародний файловий архів, який називається CTAN — Comprehensive TeX Archive Network. Основні сайти, що складають CTAN — це:

- ftp://ftp.tug.ctan.org/tex-archive
- ftp://ftp.dante.de/tex-archive/
- ftp://ftp.tex.ac.uk/tex-archive/

6 Перше знайомство

Будемо вважати, що у Вас уже встановлений[5] LaTeX 2$_\varepsilon$. Для того щоб почати працювати в LaTeX 2$_\varepsilon$, досить запустити будь-який текстовий редактор і почати набирати текст. Потім цей текстовий файл буде відкомпільований

[4] Документація уже може бути на Вашому комп'ютері. Наприклад, для MikTeX'а вона знаходиться в директорії `\texmf\doc\`

[5] Якщо LaTeX 2$_\varepsilon$ ще не встановлено, то радимо прочитати підказки по установці.

(опрацьований програмою з пакету) в приємний для ока документ. Коли LaTeX 2_ε опрацьовує вхідний текстовий файл, він очікує від нього певної будови. Так, кожний вхідний файл повинен починатися із команди[6]

`\documentclass{`*тип документа*`}`

Вона вказує на тип документа, який буде створюватися. Вибір типу документа частково визначає форматування документа. Після цього, в преамбулу (набір команд, що передують власне текстові) документа включаються команди, що впливають на оформлення документа в цілому, а також команди, що завантажують пакети, які додають нові можливості в систему LaTeX 2_ε. Для завантаження та підключення пакету використовується команда

`\usepackage{`*назва пакета*`}`

Коли вся настройка закінчена, тіло тексту починається командою

`\begin{document}`

Далі безпосередньо вводиться текст. В кінці документа додається кінцева команда

`\end{document}`

Всим, що слідує після неї, LaTeX під час транслювання (опрацювання програмою пакета) документа нехтує. Ось приклад практично найменшого вихідного текстового файла.

```
\documentclass{article}
\usepackage[russian]{babel}
\begin{document}
А тут текст, який і буде у документі.
\end{document}
```

Далі розглянемо більш складний приклад, в якому документ уже дещо структурований, тобто розбитий на підпорядковані частини. Потрібно мати

[6] В попередній версії LaTeX 2.09 документ починався з команди `\documentstyle`.

на увазі, що при компіляції той текст, що розташований в рядку після символу «%» LaTeX'ом сприймається як коментар — приватний запис для пам'яті, і тому при трансляції ним програма нехтує.

Вихідний код:

```
\documentclass[11pt]{article}
%Команда вказує, що створюється стаття,
розмір шрифта  11pt

\usepackage[ukrainian]{babel}
%Підключення підтримки української мови

\title{Назва статті}
\date{15 березня  2006 року}

\begin{document}

\maketitle
%Створюється заголовок
%Далі йде текст  статті
\section{Початок}
Текст першого розділу.

\subsection{Підрозділ}
Йде підрозділ першого розділу.

\section{Фініш}
\dots{} текст останнього розділу.

\end{document}
```

Відкомпільований документ:

Назва статті

15 березня 2006 року

1 Початок

Текст першого розділу.

1.1 Підрозділ

Йде підрозділ першого розділу.

2 Фініш

. . . текст останнього розділу.

Наведемо приклад, що показує, наскільки просто набирати математичний текст. Розглянемо код `$\int_0^t \frac {2 \nu}{\sqrt \pi} \, d \nu$`. Зверніть увагу на те, що формули розташовуються між знаками `$`. Код після трансляції дасть нам $\int_0^t \frac{2\nu}{\sqrt{\pi}}\, d\nu$. Вихідний код данної формули містить наступні команди: `\int` — це знак интеграла з нижньою границею интегрування `_0` та верхньою `^t`; `\frac` — дріб. `{2 \nu}` — перший аргумент команди `\frac` (чисельник), `{\sqrt \pi}` — другий аргумент команди (знаменник), `\,` — тонкий пропуск, `d \nu` — диференціал. Якщо подвоїти знаки `$`, то одержимо

виокремлену формулу

$$\int_0^t \frac{2\nu}{\sqrt{\pi}}\, d\nu.$$

! Зверніть увагу, що знання значень англійських слів допомагає запам'ятовуванню команд LaTeX'а.

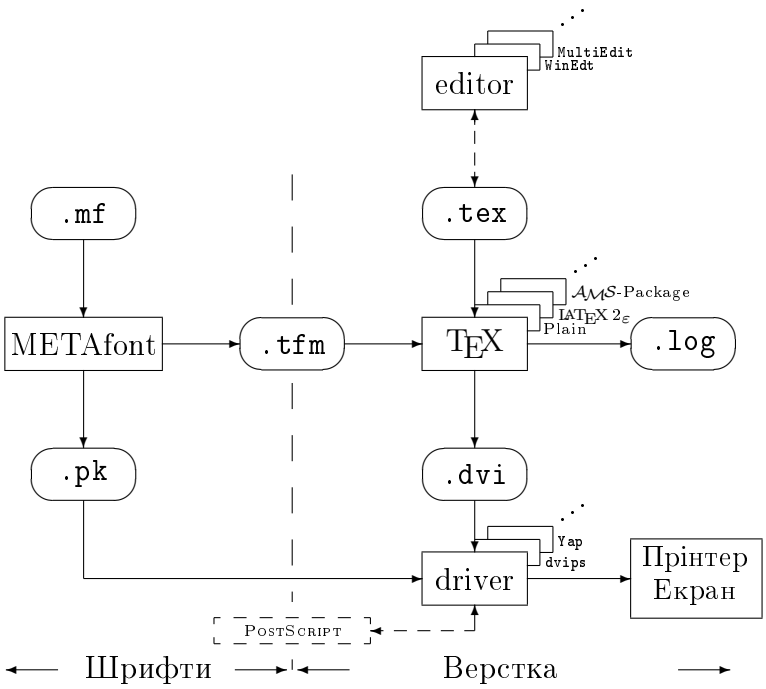


Рис. 1. Складові TEX системи

7 Як працює LATEX

Схема 1 на сторінці 17 показує, як організована робота TEX-систем (схема запозичена у Kees van der Laan). Як уже сказали, в текстовому редакторі готується текст. Потім одержаний документ зберігається у файлі з розширенням `.tex`. Далі файл транслюється в однойменний файл з розширенням `.dvi`. Одержаний вихідний файл можна переглядати на екрані, або роздруковувати на принтері, або перетворити у PostScript-файл.

Пояснимо, як TEX працює із шрифтами[1]. Очевидно, що при верстці тексту не має значення який вигляд у всіх деталях має символ. Для верстки важливо лише знати скільки місця займе той чи інший знак, тобто ширину, висоту та глибину символа, які задані відносно базової точки символа

Рис. 2. Символ «g».

[1] Детальну інформацію про шрифти можна знайти в файлі `\texmf\doc\latex\base\fntguide.*`.

(див. рис. 2). Також необхідно знати скільки місця залишити між символами та т.п. Проте, для перегляду на екрані чи для друку вихідного `.dvi`-файла, TEX-системі потрібна інформація про всі деталі кожного символа використаних шрифтів.

Зазвичай TEX працює із шрифтами METAfont. Описання розмірів шрифтів, що використовуються в TEX'і, називаются метриками шрифтів (*font metrics*) і вони описані у файлах `*.tfm`. Файли, в яких описана форма символів звичайно мають розширення `*.pk`. Зауважимо, що описання шрифта может бути дане засобами мови PostScript. Ці bitmap-образи (`.pk`) створюються для перегляду або друку на конкретному принтері з потрібною роздільною здатністю. Для розрахунку шрифта потрібна своя власна програма-драйвер. Очевидно, що при масштабуванні шрифтів їх разміри досить помножити на коефіцієнт масштабування, а вигляд шрифтів у новом `.pk`-файлі потрібно згенерувати заново.

8 Класи документів

Перше, що LATEX 2_ε повинен знати при обробці вхідного файла, це тип створюваного автором документа. Цей тип автор задає командою

`\documentclass[`*опції документа*`]{`*клас документа*`}`

Нижче перераховані вживані класи документів. Одна із цілей проекту LATEX3 полягає у збільшенні класів підтримуваних документів. Опцій документу може бути кілька, тоді вони розділяються комами. Нижче також перераховані найбільш вживані опції стандартних класів документів.

Класи документів

`article` для статей в наукових журналах, презентацій, коротких звітів, програмної документації, запрошень.

`report` для більш довгих звітів, що мають кілька розділів, невеличких книжечок, дисертацій.

`book` для справжніх книг.

`letter` для написання листів. Додаються відповідні команди форматування.

`slides` для слайдів. Використовує великі літери без зарубок[2].

Опції класів документів

`10pt`, `11pt`, `12pt` Встановлює размір основного шрифта документа. Якщо жодна із цих опцій не вказана або вказана невірна опція, то використовується розмір `10pt`.

`a4paper`, `letterpaper`... Визначає розмір аркуша. Якщо розмір не вказаний, то використовується `letterpaper`. Також можуть бути вказані `a5paper`, `b5paper`, `executivepaper` та `legalpaper`.

`draft` Режим чернетки. Місця, де програмі потрібні додаткові відомості для оформлення рядка (як правило, рядок треба перерозбити), позначаються на полях чорним прямокутником. Замість завантажуваних із файлів рисунків зображається лише рамка, яка показує місце розташування цих рисунків.

`fleqn` Виокремлені формули будуть притиснуті до лівих полів, а не відцентровані.

`leqno` Формули нумеруются зліва, а не справа.

`titlepage`, `notitlepage` Показує, чи повинен заголовок стояти на окремій сторінці (титульний аркуш). Якщо жодна з опцій не замовлена, то класс `article` не починає нову сторінку після заголовка, а `report` і `book` — починають.

`twocolumn` Примушує LaTeX 2$_\varepsilon$ набирати документ у два стовпчики[3].

[2] Для створення слайдів існує спеціальний пакет SliTeX.
[3] Для набору фрагменту текст в декілька колонок використовується пакет multicol.

`twoside, oneside` TEX може розрізняти парні та непарні сторінки. Ця опція дозволяє обирати одно- чи двустороннє виведення. Якщо не вимагається інше, класси `article` та `report` використовують одностороннє виведення, класс `book` — двустороннє виведення[4].

`openright, openany` Вказує, як повинні починатися розділи документа: тільки на правій сторінці, чи на першій доступній. Ця опція залишиться поза увагою LATEX 2_ε для класу `article`, оскільки клас `article` нічого не знає про розділи. Якщо не сказане інше, то клас `report` починає розділи із наступної сторінки, а клас `book` — з найближчої правої.

Наприклад, вхідний файл (той, що набраний в довільному текстовому редакторі, і ще не опрацьовувався), для документа LATEX 2_ε може починатися рядком

```
\documentclass[11pt,twoside,a4paper]{article}
```

Цей рядок примушує LATEX 2_ε верстати документ як *статтю*, з базовим розміром шрифта *одинадцять пунктів* та як документ для *двостороннього* друку на папері *формату A4* — шириною 210 та висотою 297 мм.

[4] Зауважте, що опція `twoside` *не* заставляє ваш принтер насправді друкувати з двох сторін ;-).

Частина II
TEXніка набору

Ближче до діла, як казав Мопасан.

Почнемо з вивчення верстки тексту. Спочатку поговоримо про логічну структуру документа. З'ясуємо, як виконується надання належного вигляду — форматування рядків та абзаців, як TEX оперує з символами. Навчимося використовувати шрифти. Познайомимося із оточеннями, блоками та навчимося створювати таблиці.

9 Логічна структура документа

Щоб допомогти читачеві орієнтуватися у його роботі, автор повинен розділити її на частини, глави, розділи та підрозділи. Взаємопідпорядкування таких частин документа називають його логічною структурою. Сам процес створення логічної структури документа називають секціонуванням документа. Як уже сказали, LATEX 2_ε підтримує розбиття документа спеціальными командами, аргументами яких є заголовок відповідної частини. Ваше діло — використовувати ці команди в належному порядку.

Клас `article` дозволяє використання наступних команд секціонування:

```
\part{...}                  \paragraph{...}
\section{...}               \subparagraph{...}
\subsection{...}            \appendix
\subsubsection{...}
```

В класах `report` та `book` додається проміжна команда `\chapter`, що починає главу, яка більше ніж розділ, який починається командою

```
\section{...},
```

і є меншим ніж частина, що починається командою

```
\part{...}
```

Оскільки глав (chapters) в класі `article` немає, то стаття досить легко додається в книгу як глава. Інтервал між розділами, нумерація та розмір шрифта заголовків встановлюються LaTeX 2_ε автоматично.

Дві із команд секціонування — особливі:

- Команда `part` не впливає на послідовність нумерації глав.
- Команда `appendix` аргумента не має. Вона просто починає нумерувати глави буквами замість цифр. Застосовується для створення додатків.

LaTeX 2_ε створює зміст, беручи заголовки розділів та номери сторінок із попереднього циклу компіляції документа[5]. Команда `\tableofcontents` виводить зміст в тому місці, де вона зустрічається. Щоб одержати правильний зміст, новий документ повинен бути оброблений LaTeX 2_ε двічі. В особливих випадках може бути необхідним і третій прохід. Коли це буде необхідним, LaTeX 2_ε вас попередить.

Всі перераховані команди секціонування існують також у варіанті із зірочкою. Такий варіант одержується додаванням `*` до імені команди. Команди із зірочкою породжують заголовки розділів, які не нумеруються і не включаються у зміст. Наприклад, команду `\paragraph{`*Даю довідку*`}` можна перетворити у команду `\paragraph*{`*Даю довідку*`}` — при цьому назва параграфа «Даю довідку» не з'явиться у змісті.

Звичайно заголовки розділів відтворюються у змісті точно в тому ж вигляді, в якому вони вводяться у тексті. Інколи це неможливо через те, що заголовок надто довгий для змісту. Те, що потрібно писати в зміст замість справжнього заголовку, в цьому випадку може вказуватися необов'язковим аргументом команди секціонування перед справжнім заголовком. Наприклад, команда

```
\chapter[Весело!]{Це~--- зовсім не нудний, ну жодним чином не нудний заголовок}
```

[5] Інформація про зміст зберігається у файлі `*.toc`. При необхідності можлива ручна правка цього файла. Наприклад, щоб зміст залишався незмінним при наступних компіляціях, треба позначити у системних властивостях файлу `*.toc` — *read only*.

створить справжній заголовок глави «Це — зовсім не нудний, ну жодним чином не нудний заголовок», а в зміст на відповідному місці поставить «Весело!»

Титульний лист документа в цілому породжується за допомогою команди `\maketitle`. Його складові частини повинні бути визначені командами

`\title{`*назва*`}`, `\author{`*автор*`}` та `\date{`*дата*`}`

до моменту виклику `\maketitle`. Якщо пропустити команду `\date`, то буде вказана поточна дата — дата трансляції. Щоб цього уникнути, потрібно задавати порожній аргумент — `\date{}`. Якщо, Ви використовуєете MikTeX, то докладний список інструкцій $\mathcal{AMS}$ з оформлення документів можна знайти у файлі `\texmf\doc\latex\amscls\instr-l.*`.

10 Пробіли та абзаци

«Порожні» символи, такі, як пробіл чи табуляція, сприймаються TEX'ом однаково, як *один* «пробіл». *Декілька послідовних* порожніх символів сприймаються як *один* «пробіл». Порожні символи на початку рядка зазвичай ігноруються, оскільки одиничне переведення на новий рядок (переведення каретки) сприймається як «пробіл».

Порожній рядок між двома рядками тексту означає кінець абзаца. *Декілька* порожніх рядків трактуються так же, як *один* порожній рядок, тобто як кінець абзаца. Нижче наводимо приклад інтерпретації TEX'ом «пробільних випадків». Справа — текст із вхідного файла, зліва — форматований вивід. Наступні приклади будуть побудовані таким же чином.

Ці всі пробіли будуть надруковані як один пробіл.
А три підряд пробіли можна одержати таким чином: Три!

```
Ці всі пробіли     будуть
надруковані як один пробіл.\\
А три підряд пробіли можна одержати
\newline таким чином: \ \ Три!
```

А цей порожній рядок починає новий абзац.

```
А цей порожній рядок починає новий абзац.
```

Як бачимо, TEX дуже слухняний і точно виконує команди.

10.1 Команди TEX'а

чутливі до регистру. Увага! Зміна малої букви на велику чи навпаки може суттєво змінити команду. Вони починаються із символа `backslash` — «\» і мають один із наступних двох виглядів:

- В команді після символу `backslash` «\» йде ім'я, що складається виключно із латинських букв. TEX розуміє, що ім'я команди закінчилося, коли після послідовності букв зустрічає пробіл, цифру чи будь-який інший символ «не-букву». Наприклад, команда `\TeX\` завершена так званим «бекслешем» — «не-буквою».
- Команда складається із «\» і точно одного спеціального символа. Прикладом може слугувати команда `\%`.

! Поширеною є помилка, коли після кінця команди не ставиться пробіл. Тоді наступний символ додається до команди, і в підсумку TEX одержує невідому собі команду. Про це при трансляції файла TEX так и каже: «`! Undefined control sequence`» — невідома команда.

Деякі команди потребують один чи декілька параметрів. Обов'язкові параметри ставляться після останнього символа імені команди між фігурними дужками — {*параметр*}. Деякі команди можна супроводжувати необов'язковими параметрами, які ставлять у квадратних дужках знову таки ж після останнього символа імені команди [*необов'язковий параметр*].

TEX нехтує пробілами, що стоять відразу після команди. Для того, щоб одержати пробіл після команди, необхідно дати команду зробити пробіл «\ » або створити порожнюю групу за допомогою пари фігуних дужок — «{}», яка завершена пробілом.

Стосовно пробілів TEX має свій каTEXізис. `Стосовно пробілів \TeX\ має свій ка\TeX ізис.`

Фундаментальне поняття TEX'а — це поняття групи. В документі групу виділяють за допомогою пари фигурних дужок «{ }», щоб виокремити частину документа в межах якої буде діяти та чи інша команда. Коли всередині більшої групи міститься менша, TEX дотримується так званого «правила більшої группи»: команди більшої группи виконуються і у внутрішній, меншій, а команди меншої групи за її межами у більшій групі не виконуються. Всередині меншої групи команда меншої групи сильніша за команду більшої.

Проілюструємо сказане за допомогою команд \it и \tiny.

Команди *похилого шрифту бувають* звичайні *та* дрібненькі. Порівнюйте.

```
Команди {\sl похилого шрифту бувають
{\rm звичайні} та {\tiny дрібненькі}}.
Порівнюйте.
```

10.2 Розриви рядків. Абзац

можна почати за допомогою команди \par. Команда \\ використовується для розриву рядків без вирівнювання по правому краю. Команда \\ допускає необов'язковий аргумент, який дає змогу змінити відстань між тими рядками, що розділені командою. Цей аргумент розуміється як відстань, що додається до міжрядкового інтервалу.

Команда \newline подібна до команди \\. Для примусового разривання рядка застосовується також команда \linebreak. Вона відрізняється від \newline тим, що розірваний рядок розтягується для вирівнювання по правому краю. Для нищення абзацного відступу є команда \noindent. Команда \parindent={...} встановлює розмір абзацного відступу.

Абзац! Алле!

Алле!

В клітку! Алле!

```
Абзац! Алле! \par
Алле!

\noindent В клітку! Алле!
```

10.3 Коментарі

створюються за допомогою символа %. Коли в процесі обробки вхідного файла TEX зустрічає символ %, він нехтує рештою поточного рядка, повернення

каретки і всі пробіли на початку наступного рядка.

Цим можна користуватися для добавляння у вихідний файл зауважень, нотаток для пам'яті та іншого, що не потрібно включати у кінцевий документ.

Знаком % можна також користуватися у випадку, коли із міркувань зручності бажано розірвати вхідний рядок (наприклад, рядок не вміщається на екрані монітора), але в цьому місці заборонено робити пробіл та переведення каретки (наприклад, це заборонено всередині імені команди)

```
Тут $\rightarrow$%
        $\leftarrow$ пробілу немає.
```

Тут $\rightarrow$% $\leftarrow$ пробілу немає.

10.4 Символи

`$ & % # _ { } ~ ^ \` використовуються TEX'ом для TEXнічних потреб. Тому їх прямий набір в `tex`-файлі приведе до того, що при трансляції файла буде повідомлення про помилку. Через це їх потрібно набирати особливим чином. У наступному прикладі наведено декілька способів.

```
Існує команда машинописного відтворення знаків \verb.$ & % # _ { } ~ ^ \ ., яка відтворює символи, що розташовані між крапками, так, мовби вони надруковані на друкарській машинці. Але це --- машинописно надруковані символи. Більш природно символи одержуються так: \$ \& \% \# \_ \{ \} \bslash
```

Існує команда машинописного відтворення знаків \verb.$ & % # _ { } ~ ^ \ ., яка відтворює символи, що розташовані між крапками, так, мовби вони надруковані на друкарській машинці. Але це --- машинописно надруковані символи. Більш природно символи одержуються так: \$ \& \% \# _ \{ \} \bslash

10.5 М'які перенесення

Під час трансляції документа TEX завжди намагається знайти найкраще розбиття тесту на рядки. Якщо він не в змозі знайти спосіб розбити текст на рядки у повній відповідності до своїх стандартів, він дозволяє одному рядку вийти за межі смуги набору вправо. Потім TEX виводить діагностику «`overfull hbox`» (переповнення строки) під час оборобки вхідного файлу. Частіше всього це трапляється, коли TEX не може знайти місце для перенесення слова. Хоч TEX і попереджує про це, такі рядки завжди легше знайти,

якщо в команді `\documentclass` використати опцію `draft`, тоді такі рядки будуть відмічені чорним прямокутником на правому полі.

Цей рядок вимагає ручного форматування. Він виходить за межі смуги набору. Так Як правило, необхідно вказати правильне перенесення в слові. TEX переносить слова автоматично, використовуючи не залежний від мови алгоритм Ліанга. Якщо після трансляції файлу читач не побачить жодного перенесення, це означає, що не включений механізм обробки переносів для конкретної мови. Для включення переносів треба зробити відповідні настройки.

М'які перенесення — це можливі місця розриву слів для перенесення. Для відмічання місць можливого разриву використовують команду `\-`. У великому документі, щоб уникнути частого відмічання можливих перенесень, доцільно описати можливі перенесення слів у преамбулі. Для цього використовується команда `\hypernation{`*спи-сок слів з пе-ре-не-сен-ня-ми*`}`. Часто список перенесень копіюється і збільшується від документа до документа. Можна очікувати, що незабаром з'явиться механізм підключения бібліотек перенесень.

10.6 Скуті одним ланцюгом

Нерозривний пробіл (символ — `~`) використовують, коли потрібно, щоб два сусідніх слова не попали на різні рядки. Така ж задача для словосполучень вирішується командою `\mbox{`*список слів*`}`.

Біллі Б. моментально запам'ятав власний код — 314159 26535 89793 23846.
Параметр *цей параметр* дуже **ва-а-а-ж-ж-ли-в-в-ий**.

```
Біллі~Б. моментально запом'ятав  власний
код~--- \mbox{314159 26535 89793 23846}.\\
Параметр \mbox{\emph{цей параметр}} дуже\\
\mbox{{\bf ва-а-а-ж-ж-ли-в-в-ий}}.
```

10.7 Пробіли між словами

Для одержання рівного правого краю тексту TEX дещо розтягує або стискує проміжки в рядку. На кінці речення він вставляє більший интервал, ніж між словами. Проміжки між реченнями розтягуються більше, ніж між словами.

Це відбувається у відповідності до традицій верстки, що прийняті в англійській мові. В українських текстах ці проміжки не повинні відрізнятися — така вітчизняна поліграфічна традиція. Для вибору такої настройки потрібно в преамбулі документа дати команду `\frenchspacing`. Допускається зміна між україномовними та англомовними стандартами безпосередньо в тексті. Для цього використовують команду `\nonfrenchspacing`.

TEX вважає, що речення закінчуються крапками, знаками питання чи оклику. Якщо крапка стоїть після букви у верхньому регістрі (великої букви), вона не вважається кінцем речення, оскільки крапки після великих букв звичайно використовуються для скорочень.

Будь-яке виключення із припущень повинно бути явно застережено автором. Знак «\» перед пробілом дає у підсумку пробіл, який не буде збільшений. Знак «~» дає пробіл, який не може збільшуватися і який, крім того, забороняє разривання в цьому місці рядка. Команда `@` перед крапкою вказує, що ця точка закінчує реченя, незважаючи на те, що вона стоїть після букви у верхньому регістрі.

```
Prof. Mr. Smith
I like KGB. What about you?
```

```
Prof.~Mr.~Smith\\
I like KGB\@. What about you?
```

10.8 Горизонтальні інтервали

Як уже відмічали LATEX автоматично визначає розміри пробілів між словами та між реченнями. Для того, щоб збільшити існуючий (чи додати новий бажаної довжини) пробіл, використовується команда `\hspace{`*довжина*`}`. Якщо такий інтервал повинен бути присутнім, навіть у випадку, коли він попадає навіть на початок чи кінець рядка, використовуйте команду `\hspace*`, а не `\hspace`.

У найпростішому випадку *довжина* — це просто число і одиниця вимірювання. Найбільш вживані одиниці перераховані нижче в таблиці.

`mm` міліметр $\approx 1/25$ дюйма
`cm` сантиметр $= 10$ mm
`in` дюйм (inch) $= 25.4$ mm
`pt` пункт $\approx 1/72$ дюйма $\approx \frac{1}{3}$ mm
`em` приблизна ширина букви ‘M’ поточного шрифта
`ex` приблизна высота букви ‘x’ поточного шрифта

Команда `\stretch{`n`}` породжує спеціальний «гумовий» пробіл. Він розтягується, заповнюючи решту місця на рядку. Коли в рядку зустрічаються дві команди `\hspace{\stretch{`n`}}`, то вони розтягуються пропорційно заданим коефіцієнтам.

Вкажемо команди, які дозволяють керувати величинами пробілів як в тексті, так і в математичних формулах. Деякі пробіли доступні лише після підключення пакету amsmath[6].

Тип пробілу	*Команда*	amsmath	*Відстань*
Подвійний math	`\qquad`		$\Rightarrow\quad\Leftarrow$
Math	`\quad`		$\Rightarrow\ \Leftarrow$
Великий	`\;`	`\thickspace`	$\Rightarrow\Leftarrow$
Середній	`\:`	`\medspace`	$\Rightarrow\Leftarrow$
Тонкий	`\,`	`\thinspace`	$\Rightarrow\Leftarrow$
Відсутність пробілу			$\Rightarrow\Leftarrow$
Від’ємний тонкий	`\!`	`\negthinspace`	$\Rightarrow\!\Leftarrow$
Від’ємний середній		`\negmedspace`	$\Rightarrow\!\Leftarrow$
Від’ємний великий		`\negthickspace`	$\Rightarrow\!\Leftarrow$

10.9 Вирівнювання по лівому краю, по правому краю та центрування

Оточення `flushleft` та `flushright` формують абзаци, в яких рядки вирівняні відповідно по лівому та правому краю. Оточення `center` дає центрований текст. Якщо місця розривання рядків автор рукопису не вказав явно, то TEX задасть їх автоматично.

[6] Пакет amsmath підключається за допомогою команди `\usepackage{amsmath}` у преамбулі документу.

11 Познайомимося із символами — цими волами абстрактного мислення

Більшість розділових знаків набираються в очевидний спосіб — натискуванням відповідної клавіші. В цьому розділі обговоримо символи, що вимагають спеціального набору. Також розповімо про використання шрифтів.

11.1 Дефіси, мінуси та тире

TEX знає чотири види тире: `-` дефіс, `--` коротке тире, `---` довге тире та `$-$` знак мінуса. Коротке тире застосовується для вказування числового проміжка. Довге тире — це звичайне тире. Зауважимо, що TEXівське тире довше від поліграфічного. Оскільки за типографськими правилами новий рядок не може починатися з тире, тому, щоб цього запобігти, потрібно ставити перед тире нерозривний пробіл — «~».

Коротке тире є прикладом лігатури, тобто випадку, коли сполучення кількох символів із вхідного файла задає один символ у кінцевому документі.

— Ледь-ледь 7–8 слоників буде, — зітхнув Петя.
— А якщо до них ще −9 пригнати?

```
--- Ледь-ледь 7--8 слоників буде,
--- зітхнув Петя.\\
--- А якщо до них ще $-9$ пригнати?
```

11.2 Лапки та дещиця інших знаків

Лапки виділяються серед інших знаків більшою каверзністю та підступністю. Тому їх розглянемо більш ретельно.

Небезпеки, що пов'язані із вжитком лапок, пов'язані з тим, що 1) вони бувають досить різних виглядів; 2) для лапок різних виглядів немає стандартизованих назв; 3) в різних мовах досить різні традиції їх вжитку; 4) при виборі лапок того чи іншого вигляду рекомендують слухати свого смаку. Автори користуються знайомими їм назвами.

Найпростішими і найпоширенішими у комп'ютерному вжитку є "комп'ютерні лапки". Поширеність полягає у простоті, а простота полягає в тому, що

це один знак, який набирається натисканням відповідної клавіші на комп'ютерній клавіатурі. Щодо краси, смаку та подібного, то, кажуть, в пристойному товаристві такі лапки не вживають.

Згідно з українською традицією рукописний текст виділяють просто лапками, тобто тими, які інші люди називають *„німецькими лапками“*. При використанні прямого друкарського шрифта українська традиція рекомедує вибирати між так званими «французькими лапками» (вживають також назву «ялинки») та “подвійними англійськими”.

В англомовному тексті виористовують “подвійні англійські” лапки та ‘одинарні англійські’.

При набиранні тексту іншою мовою рекомендуємо познайомитися із традицією вжитку лапок саме в цій мові.

І відкривні, і закривні, і німецькі, і французькі, і подвійні англійські лапки є лігатурами. Для набирання цих лігатур використовують знаки порівняння — більше, менше справа внизу на клавіатурі, апостроф — справа в середньому ряду, обернений апостроф — зліва зверху над клавішею табуляції, та кому.

Номер — №, параграф —§, ©, £, „лапки“ та «ялинки». Деякі символы є в математичних шрифтах — ♡.

```
Номер~--- \No, параграф~---\S, \copyright,
\pounds, ,,лапки‘‘ та <<ялинки>>. Деякі
символи є  в математичних шрифтах~---
$\heartsuit$.
```

11.3 Крапки, три крапки

Порівняємо крапки: просто три ... крапки та ...і ...! Останні дві команди у звичайному тексті тотожні.

Крапки використовуються і при наборі формул: $\mathbb{N} = 1, 2, 3, \cdots + \dots$

```
Порівняємо крапки: просто три ... крапки
та \dots і \ldots! Останні дві команди у
звичайному тексті тотожні.\par
Крапки використовуються і при наборі
формул: $\mathbb{N}=1,2,3,\,\dots+\dots$
```

11.4 Акценти

У розмовному вжитку акцентом називають наголос, знак наголосу. У видавничій справі акцентами називають всі можливі значки, які ставлять над

буквами. LaTeX 2_ε підтримує використання акцентів та спеціальних символів із багатьох мов. Таблиця, що наведена нижче, показує можливі акценти в застосуванні до букви «o». Зрозуміло, що на її місці повинна бути буква .... Зверніть увагу на приклад, що йде безпосередньо за таблицею.

Для того, щоб розташувати знак акцента над бувами і чи j, спочатку потрібно видалити ті крапки, що стоять над ними — це робиться командами `\i` та `\j`.

ò	`\‘o`	ó	`\’o`	ô	`\^o`	õ	`\~o`
ō	`\=o`	ȯ	`\.o`	ö	`\"o`	ç	`\c c`
ŏ	`\u o`	ǒ	`\v o`	ő	`\H o`	ǫ	`\c o`
o̩	`\d o`	o͟	`\b o`	o͡o	`\t oo`		
œ	`\oe`	Œ	`\OE`	æ	`\ae`	Æ	`\AE`
å	`\aa`	Å	`\AA`				
ø	`\o`	Ø	`\O`	ł	`\l`	Ł	`\L`
ı	`\i`	ȷ	`\j`	!‘	`!‘`	?‘	`?‘`

Ëë hôtel, naïve, élève,
smørrebrød, !‘Señorita!,
Schönbrunner Schloß Straße

```
\"E\"e h\^otel, na\"\i ve, \’el\‘eve,\\
sm\o rrebr\o d, !‘Se\~norita!,\\
Sch\"onbrunner Schlo\ss{}
Stra\ss e
```

11.4.1 Математичні акценти

В математичних формулах доступні наступні акценти:

$\tilde{A}$	`$\tilde{A}$`	$\hat{A}$	`$\hat{A}$`
$\check{A}$	`$\check{A}$`	$\bar{A}$	`$\bar{A}$`
$\breve{A}$	`$\breve{A}$`	$\vec{A}$	`$\vec{A}$`
$\acute{A}$	`$\acute{A}$`	$\dot{A}$	`$\dot{A}$`
$\grave{A}$	`$\grave{A}$`	$\ddot{A}$	`$\ddot{A}$`

11.5 Виділені слова

В рукописі, що надрукований на друкарській машинці, `важливі слова виділяються` <u>підкреслюванням</u> (команда `\underline {`*текст*`}`). В поліграфічних виданнях важливі слова виділюються *курсивом*. Команда для переключення на шрифт *виділення* — `\emph{`*текст*`}`. Очевидно, що виділення

в основному тексті, який набрано курсивом, досягається прямим шрифтом. Розглянемо декілька прикладів.

Якщо ви *виділяєте в уже виділеному тексті, то LaTeX використовує* прямий *шрифт.*

```
Якщо ви  \emph{виділяєте в уже
виділеному тексті, то \LaTeX{}
використовує \emph{прямий} шрифт}.
```

Зауважте відміни між командами *виділення* та зміни *шрифта*:

Ви маєте можливість також виділити *текст, набравши його курсивом,* шрифтом без *зарубок* або в стилі *друкарської машинки*.

```
\textit{Ви маєте можливість також
\emph{виділити} текст, набравши
його курсивом,} \textsf{шрифтом без
\emph{зарубок}} \texttt{або в
стилі \emph{друкарської машинки}}.
```

11.6 Використання шрифтів

Первісна TeX'івська система переключення шрифтів була далеко не така загальна, як та, що є в LaTeX 2_ε. Наприклад, якщо набрати текст **напівжирним** *курсивом* (`\bf напівжирним \it курсивом \normalfont`), задавши відповідні команди, то одержимо наступне: напівжирний шрифт змінився курсивом. А як одержати напівжирний курсив? В LaTeX 2_ε використана схема вибору шрифтів NFSS (New Font Selection Scheme), яка дозволяє одержати ***напівжирний курсив*** (`{\bfseries\itshape напівжирний курсив}`). Зверніть увагу, що для повернення до основного шрифта використовується команда `\normalfont`.

Такий стан речей пояснюється тим, що в першій версії LaTeX шрифт міг бути лише напівжирним чи лише курсивом чи ще лише якимось. Тому команди зміни властивості шрифта (зміна на напівжирний чи курсив) заміняли увесь шрифт. В новій схемі вибору шрифта NFSS кожний шрифт має додаткові характеристики — `family`, `series` и `shape`. Вибір типу шрифта і вибір додаткових характеристик шрифта стали незалежними (кажуть, ортогональними — мовби координати вектора). Комбінування цих характеристик (типу та додаткових характеристик) дозволяє одержати потрібний шрифт. Для

повноти викладення зауважимо, що довільний шрифт характеризується набором із п'яти атрибутів. Решта два — це внутрішнє кодування та розмір.

Не зупиняючись детально на сімействах шрифтів, наведемо кілька прикладів. Перший — приклад оточення `{\fontfamily{cmdh}\selectfont ...}`: {\fontfamily{cmdh}\selectfont Цей текст набраний шрифтом сімейства CM Dunhill.} Друга команда у прикладі `\usefont{`*size*`}{`*baselineskip*`}` вказує розміри шрифта та міжрядкового інтервалу в будь-яких одиницях довжини TEXa. Для приняття нових розмірів об'яву шрифта потрібно завершити командою `\selectfont`.

Виникає питання: чому ж лише в LaTeX 2_ε здійснили схему NFSS? Реалізація нової схеми роботи із шрифтами — це внесення суттєвих змін в TEX, до того ж, на момент внесення змін, LaTeX став стандартом de facto і його популярність тільки зростала. Відповідь вже пролунала: коли науковець створює документ, його більше цікавить логічна структура, ніж деталі оформлення. А оформлення це вже поле працівників видавництва.

Відмітимо, що використання додаткових команд форматування в тексті не заохочується. Наприклад, при наборі команд можна використовувати `машинописний шрифт` — автор так і зробив, змінивши шрифт у кожному місці, де він набирав команду. Проте коли видавництво вирішить, що потрібно використовувати, наприклад, *похилий шрифт*, то це заставить автора шукати всі місця, де він змінював шрифт — а це значний обсяг правки. Який же вихід? Засоби TEXa дозволяють створювати нові команди, зокрема, можна описати в преамбулі команду, яка вказує, який вигляд повинні мати в тексті команди. Тоді зміну шрифта досить буде виконати в одному рядку преамбули.

Нижче наведена таблиця команд зміни шрифтів. Відмітимо, що команда `\textit{`*текст*`}` рівнозначна послідовності `{\itshape` *текст*`}`.

Авторська команда	*Атрибут*	*В класі* `article`
`\textrm{..}` or `\rmfamily`	family	`cmr`
`\textsf{..}` or `\sffamily`	family	`cmss`
`\texttt{..}` or `\ttfamily`	family	`cmtt`
`\textmd{..}` or `\mdseries`	series	`m`
`\textbf{..}` or `\bfseries`	**series**	`bx`
`\textup{..}` or `\upshape`	shape	`n`
`\textit{..}` or `\itshape`	*shape*	`it`
`\textsl{..}` or `\slshape`	*shape*	`sl`
`\textsc{..}` or `\scshape`	SHAPE	`sc`
`\tiny`	size	`5pt`
`\scriptsize`	size	`7pt`
`\footnotesize`	size	`8pt`
`\small`	size	`9pt`
`\normalsize`	size	`10pt`
`\large`	size	`12pt`
`\Large`	size	`14.4pt`
`\LARGE`	size	`17.28pt`
`\huge`	size	`20.74pt`
`\Huge`	size	`24.88pt`

12 В оточенні

Оточення узагальнює поняття групи. Для верстки спеціальних видів тексту LaTeX пропонує багато оточень для різних типів форматування:

`\begin{`*назва оточення*`}` *текст* `\end{`*назва оточення*`}`

де *назва оточення* задає ім'я оточення. Оточення можна викликати всередині іншого оточення, дотримуючись певного порядку виклику та повернення:

```
\begin{зовнішне оточення}
...
\begin{внутрішне оточення}...\end{внутрішне оточення}
...
\end{зовнішне оточення}
```

В наступних розділах розповідається про найбільше вживанні оточення.

12.1 Список, перерахування та описання

Оточення `itemize` пасує для простих списків, оточення `enumerate` — для нумерованих списків, а оточення `description` — для описань.

1. Можно як завгодно вкладати одне оточення списків в друге:
   - Але це може мати безглуздий вигляд (як у цьому прикладі).
   - Або не дуже безглуздо.
2. Тому пам'ятайте:

   Нісенітниці не стануть розумнішими від розташування їх у список.

   Розумні речі, проте, цілком можна задати списком.

```
\begin{enumerate}
\item Можно як завгодно вкладати
одне оточення списків в друге:
\begin{itemize}
\item Але це може мати безглуздий
вигляд (як у цьому прикладі).
\item[-] Або не дуже безглуздо.
\end{itemize}
\item Тому пам'ятайте:
\begin{description}
\item[Нісенітниці] не стануть розумнішими
від розташування їх у список.
\item[Розумні] речі, проте, цілком
можна задати списком.
\end{description}
\end{enumerate}
```

12.2 Цитати та вірші

Оточення `quote` використовується для набору цитат, важливих фраз та прикладів, воно не створює відступу на початку рядка.

Типографське правило для довжини рядків:

> Жоден рядок не повинен мати більше 66 символів.
>
> От чому LaTeX робить поля сторінок такими широкими.

Тому в газетах часто застосовують набір у декілька колонок.

```
Типографське правило для довжини рядків:
\begin{quote}
Жоден рядок не повинен мати більше~66
символів.

От чому \LaTeX{} робить поля сторінок
такими широкими.
\end{quote}
Тому в газетах часто застосовують
набір у декілька колонок.
```

Існує ще два схожих оточення: `quotation` та `verse`. Оточення `quotation` використовується для набору більш довгих цитат, які охоплюють декілька абзаців — це оточення начинає новий абзац з нового рядка.

Оточення `verse` використовують для набору віршів, коли важливі розривання рядків. Рядки розділяються за допомогою `\\` наприкінці рядка та

за допомогою порожнього рядка після кожної строфи. Команда `\\*` також починає новий рядок, але вона забороняє перенесення наступного рядка на наступну сторінку.

Я знаю лише один вірш напам'ять:

Мама мила мене мила
Поки вистачило мила.

```
Я знаю лише один вірш напам'ять:
\begin{flushleft}
\begin{verse}
Мама мила мене мила\\*
Поки вистачило мила.\\
\end{verse}
\end{flushleft}
```

12.3 Буквальне відтворення

Текст, що розташований між `\begin{verbatim}` та `\end{verbatim}` буде надрукований безпосередньо, так, мовби він був набраний на друкарській машинці, з усіма пробілами та переведеннями каретки, без виконання якої б то не було команди LaTeX.

Всередині абзацу подібну функцію виконує команда `\verb+`*текст*`+` Тут «+» — це тільки приклад символа-обмежувача. Ви маєте змогу в якості символа-обмежувача використовувати будь-який символ, крім букв, «*» та пробілу. Символ-обмежувач не повинен зустрічатися у відтворюваному тексті, а сам текст повинен уміститися в один рядок. Багато прикладів, що стосуються LaTeX 2_ε в цьому тексті набрані саме цією командою.

Команда `\ldots` ...

```
Команда \verb| \ldots| \ldots
```

Згадані команда та оточення існують також у варіанті із зірочкою — є команда `\verb*` і є оточення `verbatim*`. Дія команди та оточення із зірочкою відрізняється від своєї пари без зірочки тим, що пробіли замінюються підкреслюванням.

Заборонене використання оточення `verbatim` та команди `\verb` всередині параметрів інших команд.

Пакет alltt вводить оточення із цією ж назвою. Воно схоже на `verbatim`.

Відміна полягає лише в тому, що \, { та } зберігають своє звичайне призначення — бути спецсимволами.

Коли часто використовується команда \verb, то виникає бажання полегшити свою працю. За допомогою пакета shortvrb можливо командою \MakeShortVerb{*symbol*} вказати символ, який буде визначати початок та кінець буквального відтворення. Команда \DeleteShortVerb{*symbol*} відновлює звичайне використання символа.

Тепер \команда робить те ж, що і \команда.

```
\MakeShortVerb{\/} Тепер \verb/\команда/
робить те ж, що і /\команда/.
```

12.4 Таблиці

Для верстки таблиць використовують оточення tabular. В таблиці можна вказувати як горизонтальні так і вертикальні лінії. LaTeX визначає ширину стовпчиків автоматично.

Аргумент *специфікація* оточення \begin{tabular}{*специфікація*} визначає формат таблиці. Використовуйте специфікації l (left) для стовпчиків тексту, що вирівняні вліво, r (right) для тексту, що повинен бути вирівняний вправо і c (center) для центрованого тексту, p{*ширина*} для стовпчика, що містить вирівняний текст з перенесенням рядків, і | для вертикальної лінії. Опція p{*ширина*} вирівнює текст в боксі по верхньому рядку, якщо взяти опції m (middle – середина) чи b (bottom – дно), то текст вирівнюється по середині рядка і по низу рядка відповідно.

Всередині оточення tabular знак «&» вказує на перехід до наступного стовпчика, команда \\ починає новий рядок, а \hline вставляє горизонтальну лінію. Лінії проводяться також операторами \vline — на повну висоту та глибину рядка, а оператором \cline{*i*–*j*} проводиться горизонтальна лінія від колонки з номером i до колонки с номером j.

54	шістнадцяткове
124	вісімкове
1010100	двійкове
84	десяткове

```
\begin{tabular}{|r|l|}   \hline
54 & шістнадцяткове \\ 124 & вісімкове \\
1010100 & двійкове \\ \hline \hline
84 & десяткове \\ \hline
\end{tabular}
```

Запрошуємо в абзац, що у рамочці. Сподіваємося вам усім тут сподобається.

```
\begin{tabular}{|p{5.55cm}|} \hline
Запрошуємо в абзац, що у рамочці.
Сподіваємося вам усім тут сподобається.\\
\hline
\end{tabular}
```

Роздільник стовпчиків можна задати конструкцією `@{...}`. Ця команда видаляє пробіл між стовпцями і замінює його на те, що розташоване між фігурними дужками. Одне из частих застосувань цієї команди показане нижче, в прикладі вирівнювання стовпчика чисел по десятковій точці. Конструкцію `@{}` можливо також використовувати для заглушення в таблиці кінцевих пробілів:

немає ні початкового ні кінцевого пробілів

```
\begin{tabular}{@{} l @{}} \hline
немає ні початкового ні кінцевого пробілів\\
\hline \end{tabular}
```

є початковий і є кінцевий пробіли

```
\begin{tabular}{l} \hline
є початковий і є кінцевий пробіли\\
\hline \end{tabular}
```

В LaTeX не існує способу вирівнювання числових стовпців по десятковій точці. Але можна «обманути» TeX — створити два стовпчики: перший є вирівняною вправо цілою частиною, а другий — вирівняною вліво дробовою. Команда `@{.}` в рядку `\begin{tabular}` замінює нормальний пробіл між стовпчиками просто на «.», створюючи ефект одного стовпчика, що вирівняний по десятковій точці. Не забудьте замінити у ваших числах точку на раздільник стовпчиків (`&`)! Мітку стовпчика можна розташувати над нашим числовим «стовпчиком» командою `\multicolumn`:

Вираз із π	Значення
π	3.1416
π^{π}	36.46
$(\pi^{\pi})^{\pi}$	80662.7

```
\begin{tabular}{c | r @{.} l}
Вираз із $\pi$       &
\multicolumn{2}{c}{Значення} \\
\hline
$\pi$                & 3&1416  \\
$\pi^{\pi}$          & 36&46   \\
\cline{2-3}
$(\pi^{\pi})^{\pi}$ & 80662&7 \\
\end{tabular}
```

Зверніть увагу на оператор `\multicolumn{`n`}{`col`}{`$text$`}`, який створює вічко, що складається із тексту {$text$}, що займає {n} колонок та позиціонована у відповідності із специфікацією {col}. Коли потрібно змінити позиціонування конкретного вічка, то досить опцію кількості колонок зробити рівною 1 ({n}= 1) та визначити позиціонування в опції {col}.

12.5 Блоки

TEX вибудовує сторінки, пересуваючи блоки абзаців, малюнків, таблиць. Абзаци формуються із слів, а слова із букв. Спочатку кожна буква є маленьким блоком, який приклеюється до інших букв, утворюючи слово. Слова склеюються із іншими словами за допомогою спеціального еластичного клею, який може як розтягуватися так і стискатися, рівно настільки, щоб точно заповнити рядок.

Потрібно визнати, що це досить спрощена версія того, что відбувається насправді, але суть в тому, що TEX завжди має справу з блоками та клеєм. Не тільки буква може бути блоком. Ви можете помістити в блок практично все, що завгодно. Блоки можна використовувати для побудови інших блоків. Кожний блок потім опрацьовується TEXом, незалежно від розмірів, як монолітна складова. Як уже згадували на сторінці 17, найелементарніший блок — буква характеризується шириною, висотою та глибиною (див. мал. 2 на стор. 17).

Для прикладу, оточення `tabular` чи `includegraphics` (використовується для вставляння ілюстрацій) формують блок. Це означає, що ви маєте змогу легко розташувати поряд дві таблиці чи ілюстрації. Тільки необхідно переконатися, що їх загальна довжина не перевищує `\textwidth` (ширини тексту).

Ви також маєте змогу упакувати будь-який абзац у блок як командою `\parbox[`$поз$`]{`$ширина$`}{`$текст$`}`, так і оточенням `\begin{minipage}[`$поз$`]{`$ширина$`}`$текст$ `\end{minipage}`

Параметр {*поз*} може бути однією з букв `c`, `t` або `b`[7], вказуючи, яким повинно бути вертикальне вирівнювання блоку відносно базової лінії оточуючого тексту. Параметр {*ширина*} має аргументом ширину блока. Основна відміна оточення `minipage` від `\parbox` полягає в тому, що всередині `\parbox` заборонено використання команд та оточень, тоді як всередині `minipage` дозволено практично все.

В той час, як `\parbox` упаковує цілий абзац, разбиваючи рядки та інше, існує клас блокових команд, що працюють лише на горизонтально розташованному матеріалі, тобто лише з одним рядком. З однією з таких команд ми вже знайомі. Це команда `\mbox`, яка упаковує послідовність блоків, що можна використовувати для запобігання початку нового рядка. Оскільки розміри блоків, що стоять у рядку не мають значення, то упаковщики горизонтальних блоків є досить потужним інструментом. Наприклад, сторінку, що створена оточенням `minipage`, дуже легко взяти у рамку.

Узагальненням команди `\mbox` є більш гнучка команда `\makebox[`*ширина*`][`*поз*`]{`*текст*`}`, яка точно так же формує блок з тією відміною, що у неї є декілька параметрів. Параметр {*ширина*} визначає ширину результуючого блоку так, як його буде видно іззовні, тобто можна задати ширину, що відрізняється від реальної[8]. Крім виразів довжини, можно використовувати параметри `\width`, `\height`, `\depth` та `\totalheight`. Вони встановлюються рівними значенням, одержаним вимірюванням розмірів блока, що складений із вхідного {*тексту*}.[9] Параметр [*поз*] приймає однобуквенні значення: **c** – центрувати, **l** – притиснути вліво, **r** – притиснути вправо або **s** – рівномірно заповнити блок текстом.

Команда `\framebox` працює точно так, як і `\makebox`, але ще додатково креслить рамку наколо текста. Короткий аналог — команда `\fbox`.

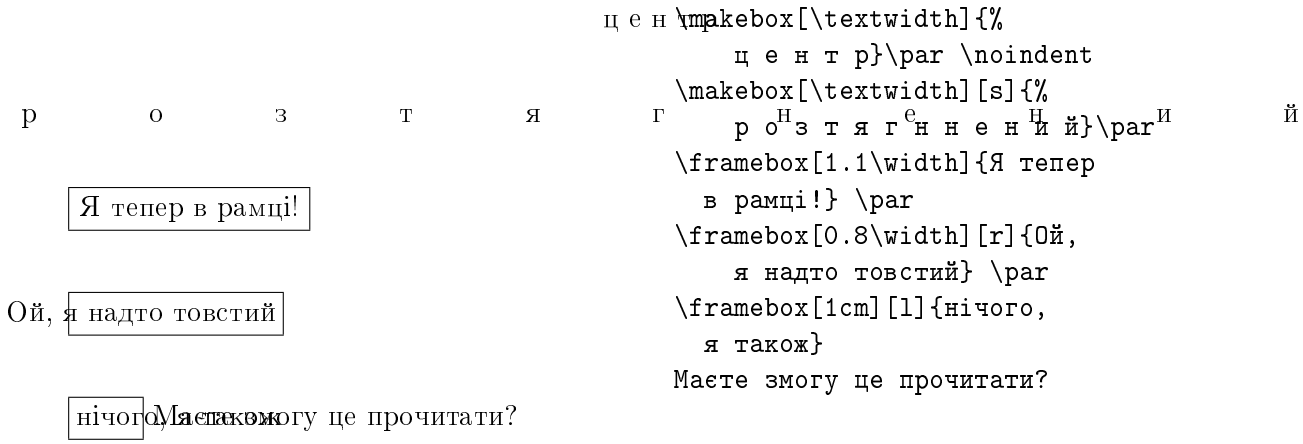
Наступний приклад показує можливості використання команд `\makebox`

[7] `c`=center, `t`=top, `b`=bottom

[8] Це означає, що вона може бути меньшою ніж матеріал всередині блока. Можна навіть встановити її рівною 0pt, так що текст всередині блока верстається, взагалі не впливаючи на оточуючі блоки.

[9] Ширина, висота, глибина та загальна висота (висота плюс глибина) тексту, відповідно.

та \framebox.



Тепер, коли ми керуємо горизонталлю, очевидний наступний крок — вертикаль. Ніяких проблем. Для цього використовуємо команду \raisebox {*зсув*}[*глибина*][*висота*]{*текст*}, яка дозволяє задати вертикальні характеристики блока. В перших трьох параметрах можна використовувати параметри \width, \height, \depth та \totalheight, що збігаються з дійсними розмірами аргумента {*текст*}, щоб задати нові розміри для розташування блока TEXом.

12.6 Рухомі об'єкти

В наш час більшість публикацій містить помітну кількість ілюстрацій та таблиць. Ці елементи вимагають спеціального поводження з ними, оскільки вони не можуть бути розірвані між сторінками. Один із можливих підхідів полягає в тому, щоб починати нову сторінку кожний раз, коли зустрічається ілюстрація чи таблиця, надто велика, щоб уміститися на поточній сторінці. Цей підхід привів би до того, що сторінки залишалися б частково порожніми, що є ні естетично ні економічно прийнятним.

Для розв'язування цієї проблеми будь-яку ілюстрацію чи таблицю, що не уміщається на поточній сторінці, можна заставити «плавати», пересуватися в процесі заповнення текстом поточної сторінки на наступну. LATEX пропонує для рухомих об'єктів два оточення, — одне для таблиць і одне для ілюстрацій.

Поведінка цих оточень нічим, крім підпису, не відрізняються. Щоб повністю використати їх переваги, важливо хоч приблизно уявляти, як LaTeX опрацьовує рухомі об'єкти. Інакше вони можуть стати джерелом розчарування із-за того, що LaTeX розташовує їх не там, де їх бачить автор.

Спочатку розглянемо команди, які пропонує LaTeX для рухомих об'єктів.

Будь-який матеріал, що включений в оточення `\figure` або `\table`, вважається рухомим. Обидва оточення `\begin{figure}[`*специфікація розташування*`]` та `\begin{table}[`*специфікація розташування*`]` мають необов'язковий параметр, який називають `[`*специфікацією розташування*`]`. Цей параметр використовується для вказівки LaTeX, куди в першу чергу бажано помістити рухомий об'єкт. `[`*Специфікація розташування*`]` конструюється шляхом збирання в рядку *ключів розташування рухомого об'єкта*. Див. таблицю 1.

Табл. 1. Ключі розташування рухомого об'єкта

Ключ	Дозволяє помістити об'єкт ...
`h`	*тут же*, в тому самому місці тексту, де він з'явився. Звичайно використовується для маленьких об'єктів.
`t`	*наверху* сторінки.
`b`	*внизу* сторінки.
`p`	на *спеціальній сторінці*, яка містить лише рухомі об'єкти.
`!`	не враховувати більшість внутрішніх параметрів, які могли б заважати розташуванню цього об'єкта.

Таблицю можна почати наступним рядком: `\begin{table}[!hbp]`. Специфікація розташування `[!hbp]` пропонує першою здійснити спробу LaTeX розташувати таблицю прямо на вказаному місці (`h`), другою спробою пропонується розташувати внизу тієї ж сторінки (`b`), а при двох невдачах розташувати на виділеній сторінці (`p`), і все це навіть у випадку, коли вигляд утворення не відповідає стандартам (`!`). Якщо ніякої специфікації розташування автор не задав, стандартні класи вважають за недомовленістю специфікацію `[tbp]`.

До розташування рухомих об'єктів LaTeX відноситься вельми шанобливо, намагаючись не порушити гармонію формування сторінки. Тому-то він і

використовує велику кількість параметрів. Наприклад, в стильовому файлі зарані визначена максимальна кількість рухомих об'єктів, які можна розташувати на одній сторінці. Специфікація розташування `!` послаблює вимоги та полегшує розташування рухомих об'єктів.

LaTeX розташовує кожен із зустрінутих рухомих об'єктів у відповідності до заданої автором специфікації. Якщо об'єкт неможливо помістити на поточній сторінці, він відкладається і ставиться в чергу ілюстрацій чи в чергу таблиць[10]. Коли починається нова сторінка, LaTeX перевіряє, чи можна заповнити спеціальну сторінку рухомими об'єктами із черг. Якщо ні, то перший об'єкт із кожної черги вважається таким, що тільки що з'явився в тексті: LaTeX знову пробує розташувати їх у відповідності до їх специфікацій (за виключенням «h», що вже неможливо). Нові рухомі об'єкти, що зустрічаються в тексті, ставляться у хвіст відповідної черги. LaTeX зберігає порядок, в якому зустрілися рухомі об'єкти відповідного типу. Тому обтяжлива ілюстрація, яку не вдається помістити на сторінці, відштовхує решту ілюстрацій в кінець документа.

Для виведення усіх накопичених в черзі об'єктів слугує команда `\clearpage`.

Після пояснення цих механізмів лишається додати декілька зауважень про оточення `table` та `figure`. Командою `\caption{`*текст підпису*`}` можна задати підпис до об'єкта. Збільшуваний номер и слово «Рисунок» чи «Таблиця» створюються LaTeX автоматично. При недомовленості в підписі в якості разділювача використовується двокрапка. Пакет `caption2` вводить кілька нових команд, що дозволяють налагодити формат підпису. Наприклад, для зміни розділової двокрапки на звичну крапку використовується команда `\renewcommand{\captionlabeldelim}{.}`

Дві команди `\listoffigures` та `\listoftables` працюють подібно до команди `\tableofcontents`, друкуючи, відповідно, список ілюстрацій чи таблиць. В цих списках заголовки повторюються повністю. Якщо ви використо-

[10]Ці черги підкоряються дисципліні FIFO: „First In — First Out“, що українською мовою означає, хто до нас першим прийде, той першим і вийде.

вуєте довгі заголовки, то ви повинні надати їх короткий варіант для включення у списки. Цього досягають розташуванням короткого варіанта у квадратні дужки після команди `\caption`. Наприклад, `\caption[Короткий]{Ддддоооввввжжжеееелл`

За допомогою команд `\label` та `\ref` можна створювати посилання із тексту на рухомий об'єкт. Посилання на номер сторінки створюється командою `\pageref`. Подробиці можна знайти у розділі 13 на стор. 46.

Наступний приклад креслить квадрат і вставляє його в документ. У такий засіб можна залишити в документі місце під зображення, які будуть вставлені пізніше.

```
Рисунок~\ref{white} є  прикладом Поп-Арта.
\begin{figure}[!hbp]
\makebox[\textwidth]{\framebox[2cm]{\rule{0pt}{2cm}}} \caption{Два
на два сантиметри} \label{white}
\end{figure}
```

Рисунок 3 є прикладом Поп-Арта.

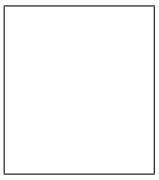


Рис. 3. Два на два сантиметри

В цьому прикладі LaTeX буде *вельми кріпко* (`!`) намагатися розташувати ілюстрацію прямо *по місцю* (`h`)[11]. Коли це неможливо, він спробує розташувати її *внизу сторінки* (`b`). Коли йому не вдасться розташувати ілюстрацію на поточній сторінці, він з'ясує, чи можна створити сторінку рухомих об'єктів, що містить у собі нашу ілюстрацію, можливо, деякі талиці із черги таблиць. Якщо для окремої сторінки материалу ще не накопичилось, LaTeX починає нову сторінку і знову розглядає можливість розташувати ілюстрацію, немов вона з'явилася тільки що в тексті.

Для створення от таких рухомих об'єктів, що обтікаються текстом, використовується пакет floatflt. Цей пакет визначає оточення `floatingfigure` з опцією {*ширина*}. Глибина об'єкта обчилюється автоматично. Пакет floatflt

[11] За умови, що черга ілюстрацій порожня.

повністю сумісний із стандартним оточенням `figure`. За недомовленістю рухомі об'єкти «в оборку» разташовуються у відповідності з парністю сторінок: на непарних сторінках — справа, на парних — зліва. Проте, можна в необов'зковому параметрі вказати притискування до правого краю [*r*] чи притискуання до лівого [*l*].

Для створення таблиць «в оборку» потрібно брати оточення `floatingtable` {*таблиця*}, де {*таблиця*} і є набрана стандартными засобами таблиця, що обтікається

English	Russian
top	верх
here	тут
bottom	низ

Табл. 2. Таблиця справа

Потрібно відмітити, що пакет floatflt не сумісний із режимом twocolumn. До того ж малюнок, що обтікається, неможливо розташувати в тому абзаці, яким починається сторінка. Якщо при використанні цього пакета буде проігнованим текст, що йде після оточення `floatingfigure`, то потрібно пересунути саме оточення на декілька абзаців вперед чи назад.

13 Автоматизація

13.1 Перехресні посилання

В книгах, звітах та статтях часто використовуються перехресні посилання на ілюстрації, таблиці та окремі частини тексту. Для створення посилань LaTeX надає наступні команди:

`\label{`*мітка*`}`, `\ref{`*мітка*`}` и `\pageref{`*мітка*`}`,

де *мітка* — вибраний користувачем ідентифікатор. LaTeX замінює `\ref` номером розділу, підрозділу, ілюстрації, таблиці чи рівняння, де була використана відповідна команда `\label`. Команда `\pageref` друкує номер сторінки, на якій зустрілася команда `\label`. Так же, як і у випадку із заголовками розділів, тут також використовують номери, що створені попереднім транслюванням.

Підкреслимо, що команди посилань не знають, на що саме вони посилаються. `\label` просто зберігає останній номер із автоматично згенерованих номерів у робочому файлі. Разширення цього файла — `*.aux`. Подивіться його за допомогою звичайного текстового редактора і вам стане зрозумілим, чому іноді потрібно транслювати `tex`-файли двічі.

Посилання на цей розділ має такий вигляд: «див. розділ 13.1 на стор. 47.»

```
Посилання  на цей розділ~\label{sec:this}
має такий вигляд: <<див.
розділ~\ref{sec:this} на
стор.~\pageref{sec:this}.>>
```

13.2 Винесені виноски[12]

Щоб зробити виноску, використовують команду

`\footnote{`*текст виноски*`}`

яка друкує виноску внизу поточної сторінки. При цьому виноски нумеруються підряд. Як правило, нумерація виносок починається спочатку, коли починається нова глава. Команда виноски завжди повинна ставитися після слова, до якого вона відноситься. Якщо після слова повинен бути розділовий знак, то спочатку робиться виноска, і тільки потім ставиться потрібний знак.

Користувачі LaTeX часто використовують виноски[13].

```
Користувачі \LaTeX{}
часто використовують виноски\footnote{%
Це~--- знесена виноска.}.
```

У команд виноски є необов'язковий аргумент — число, яке буде номером виноски[132335546]. Переконаємося, що нумерація виносок при цьому не порушується[14].

Припустимо, що потрібно зробити виноску всередині деякого блока. Спочатку вказується місце виноски командою `\footnotemark`. Після закінчення

[12] Текст що зноситься вниз — знесена виноска :-).

[13] Це — знесена виноска.

[132335546] Наприклад, так.

[14] Таки так! LaTeX проігнорував цей номер.

блоку потрібно дати текст виноски командою `\footnotetext{`*текст виноски*`}`. Для підказки бажаного номера виноски, його потрібно набрати, як необов'язковий аргумент в обох командах.

Правильна виноска[a] в оточенні `minipage`. У цієї виноски[15] текст в оточенні. А це трійка виносок:

Г[16]-П[17]=Г[18]за вітром

[a]Це — правильна виноска.
[a]В оточенні.

```
\begin{minipage}{5 cm}
Правильна виноска\footnote{Це~---
правильна виноска.} в оточенні
\texttt{minipage}. У цієї виноски%
\footnotemark\footnotetext{В оточенні.}
текст в оточенні. А це
трійка виносок:
\[\fbox{Г\footnotemark -П\footnotemark
=Г\footnotemark за вітром} \]
\end{minipage}
\footnotetext{Всі виноски підраховані.}
\footnotetext{Здогадайтесь
чому так.}
```

Пояснимо, що відбулося насправді. За командою `\footnotemark` лічильник виносок на сторінці збільшується на одиницю і виводиться у тому місці тексту, де ми дали команду. Друга, доповнююча команда `\footnotetext{`*текст виноски*`}` просто виводить у виноску `{`*текст виноски*`}` користуючись при цьому поточним значенням лічильника. До чого це може призвести видно із приклада.

Який ... Який же зараз номер виноски[19]? Невже неможливо знайти управу на ці номери? Можна. Є така команда!

`\addtocounter{`*footnote*`}{`*число, яке додається до лічильника*`}`

Нумо. Випробуємо [1020]. Ну, тоді повернемо все як було[21].

```
Нумо \addtocounter{footnote}{+1000}.
Випробуємо \footnote{Працює!}
\addtocounter{footnote}{-1000}. Ну,
тоді повернемо  все як
було\footnote{Отак!}.
```

Для створення виноски в заголовці потрібно використати необов'язковий

[18]Всі виноски підраховані.
[18]Здогадайтесь чому так.
[19]Ф-ф-у-у-х. Правильний!
[1020]Працює!
[21]Отак!

аргумент заголовка, який з'являтися у змісті, колонтитулах та подібному. Уявіть, який вигляд буде мати виноска у змісті. Уявили? От тому виноску потрібно використовувати в обов'язовому аргумені, і дублювати заголовок у необов'язковому аргументі вже без виноски.

13.3 Бібліографія

Коли пишеться стаття, реферат, про книгу уже й не говоримо, не уникнути посилань на джерела інформації LaTeX надає засоби для створення бібліографії. Для кожного джерела автор повинен придумати {*маркер*}, за допомогою якого він буде робити необхідні посилання на це джерело. Посилання створюється командою

`\cite{`*маркер*`}`

Для створення бібліографії використовується оточення `thebibliography`. Кожний елемент оточення починається з команди

`\bibitem{`*маркер*`}`

Нумерація елементів бібліографії здійснюється автоматично в тому порядку, в якому вони перераховані в бібліографічному оточенні. Параметр після команди `\begin{thebibliography}` встановлює максимальну ширину номерів. У наступному прикладі `{99}` вказує LaTeX, що жоден із номерів элементів бібліографії не буде більше 99.

У праці А.Д. Александрова було висловлене припущення [1, с. 231] про те, що ... Погорєлов [2] довів, що ...

```
У праці А.Д.~Александрова було висловлене
припущення~\cite[с. 231]{alek48} про те,
що \ldots Погорєлов~\cite{pog69} довів,
що \ldots
\begin{thebibliography}{99}
\bibitem{alek48}
Александров А.Д. Внутренняя геометрия
выпуклых поверхностей.
{\em М.: Гостехиздат, 1948.}
\bibitem{pog69}Погорелов А.В. Внешняя
геометрия выпуклых поверхностей.
{\em М.: Наука, 1969.}
\end{thebibliography}
```

Література

[1] Александров А.Д. Внутренняя геометрия выпуклых поверхностей. *М.: Гостехиздат, 1948.*

[2] Погорелов А.В. Внешняя геометрия выпуклых поверхностей. *М.: Наука, 1969.*

13.4 Показчики

Шукаючи потрібну інформацію часто приходиться використовувати предметний показчик. Предметний показчик є в більшості поважних книг. Проте, такий показчик настільки просто створюється, що Ви маєте змогу витворити таке навіть у курсовій роботі. Показчик створюється автоматично за допомогою команд LaTeX та супроводжуючої програми `makeindex`[22]. Розглянемо тільки базові команди породження показчика.

Зміст показчика створюється командами

```
\index{ключ показчика}
```

де {*ключ показчика*} є елементом показчика. Команди показчика пишуться в тому місці текста, на яке цей елемент повинен показувати. Таблиця 3 пояснює синтаксис аргумента {*ключ показчика*} декількома прикладами.

Табл. 3. Приклади синтаксиса ключів показчика

Приклад	**Вид показчика**	**Коментар**
`\index{перенесення}`	перенесення, 1	Звичайний елемент
`\index{перенесення!м'яке}`	м'яке, 3	Підпорядкований елемент
`\index{перенесення@\textsl{перенесення}}`	*перенесення*, 2	Форматований ключ
`\index{перенесення|textbf}`	перенесення, **3**	Форматована сторінка

Для підклюіення описаних можливостей LaTeX у преамбулі документа повинен завантажуватися пакет пакет makeidx. Щоб при компіляції LaTeX формував список ключів показчиків потрібно додати декларацію `\makeindex`:

```
\usepackage{makeidx}
\makeindex
```

Протягом обробки документа, список ключів показчиків записується у файл з розширенням `.idx`. Якщо видалити декларацію `\makeindex`, то такий

[22]На системах, що не підтримують довгі імена файлів, программа може називатися `makeidx`.

файл створюватися не буде. Замість нього буде використовуватися раніше сформований показчик.

Щоб включити в документ сформований показчик, дають команду:

```
\printindex
```

Проте, перед тим, як буде надрукований предметний показчик, потрібно ще раз відкомпілювати `idx`-файл за допомогою згаданої програми `makeindex`[23]. Під час виконання програми створюється лог-файл із розширенням `ilg`.

Під час перевірки текста і предметного показчика зручно використовувати пакет showidx. Цей пакет друкує всі елементи показчика на лівому полі текста.

13.5 Нові команди, оточення та пакети

TEX надає чудові засоби для виконання об'ємної однотипної роботи. В найбільш простому випадку може набриднути набирати багато разів один і той же вираз. Тоді можна створити нову команду, яка зменшить кількість символів, які потрібно набрати. Коли ж приходиться часто набирати однотипні оточення чи команди форматування, то варто визначити власне оточення. А якщо до цього додати створення власного макету документа, то зручніше всього все це описати в стильовому файлі.

13.5.1 Нові команди

Нехай часто приходиться писати «...як писав вельмишановний Бурундук Бурундукович Трисмуговий у своєму монументальному творі...». Тоді зручно[24] визначити команду `\vbbt`:

[23] Як запускається `makeindex` на Вашому комп'ютері, Ви повинні з'ясувати самостійно. Як правило, досить дослідити меню оболонки, за допомогою якої Ви транслюєте LaTeX документи. В оболонці `Winedt` в пункті меню **Accessories** потрібно вибрати **Makeindex**.

[24] Як і що зручно — вирішуєте Ви.

```
\newcommand{\vbbt}{многоуважаемый
Бурундук бурундукович Трисмуговий}
\dots як це робив \vbbt{} завжди\\
Хейо, \vbbt    !
```

. . . як це робив вельмишановний Бурундук Бурундукович Трисмуговий завжди
Хейо, вельмишановний Бурундук Бурундукович Трисмуговий!

Припустимо, що вам необхідно підготувати деякий формуляр. В формулярі потрібно створювати проміжи-лінійки різної довжини для анкетних даних. Для цієї мети знадобиться команда `\rule{`*довжина*`}{`*висота*`}`, яка створює лінійку заданої довжини та товщіни.

```
\newcommand{\dwidth}[1]{\rule{#1}{0.3pt}}
\\ \dwidth{.4\textwidth}\\
П.І.Б. \dwidth{10 em} Вік \dwidth{2 em}
```

_____
П.І.Б. _____ Вік ____

➢Вправа 13.1

Запишіть команду для набору конструкції $\frac{\partial \dots}{\partial \dots}$ частинної похідної виразу по деякій змінній. Спробуйте виконати вправу самостійно, а потім подивіться відповідь, яка наведена нижче.

```
\newcommand{\pl}[2]{$\frac%
{\partial #1}{\partial #2}$}
\pl{f(x^2-z^3)}{z}, \pl{f}{x}
```

$\frac{\partial f(x^2-z^3)}{\partial z}$, $\frac{\partial f}{\partial x}$

➢Вправа 13.2

Який недолік має наведене вирішення вправи 13.1? Спробуйте набрати, для прикладу, вираз $F\left(\frac{\partial f}{\partial x}, \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z}\right)$.

Підсумуємо. Для створення власних команд використовується конструкція

`\newcommand{`*назва нової команди*`}[`*кількість аргументів 2-9*`]{`*визначення*`}`

Кількість аргументів не повинна перевищувати 9. Коли TEX при трансляції документа зустрічає таку команду, то він виконує звичайну підстановку в текст на місце цієї команди. Інколи це може дати небажані эффекти, що пов'язані з непродуманим описанням команди. Тому при появі різного роду

неприємностей варто починати пошук помилок з перевірки нововизначених команд. Приклад поширеної помилки, коли забувають задати групу у визначенні, і одержують такий ефект:

Така я Яна ясна. Ясно, треба так:
и Це правильно!

```
\newcommand{\oops}[1]{\huge #1}
Така \oops{я Яна} ясна.
Ясно, треба так: \normalsize \\
\renewcommand{\oops}[1]{{\huge #1}}
И \oops{це} правильно!
```

Команда `\renewcommand` перевизначає команду, яка уже існує. якщо спробувати використовувати команду `\newcommand`, то TEX повідомить про помилку. Якщо ж заміна, коли команда уже визначена, небажана, то потрібно використати для її захисту команду `\providecommand`.

Може читач уже помітив помилку при визначенні частинної похідної. Дамо йому час подумати, розташувавши відповідь у виносці[25]. Можливо, звичайно, викинути символи $ із визначення команди, але тоді їх прийдеться ставити в тексті. Проте, в LATEX 2_ε все уже передбачено. Аргумент команди `\ensuremath{`*math code*`}` завчасно буде входити в документ в математичному режимі. Про це потурбується LATEX 2_ε.

Як приклад визначимо команду `\xvec` для запису векторів, таких як $x_1, \dots, x_n$. В такому виразі можна виділити дві можливі змінні — назву вектора та його вимірність. Отже будемо визначати команду з двома параметрами. Проте, якщо імена векторів змінюються часто, то вимірність змінюється суттєво рідше. Зробимо так, щоб вимірність була необов'язковим аргументом. LATEX дозволяє визначати тільки один необов'язковий аргумент, його номер при визначенні завжди — # 1. І ніяких варіантів. При визначенні значення за недомовленістю вказується у квадратних дужках після вказівки кількості аргументів. Ось приклад

Візьмемо у праву руку $x_1, \dots, x_k$, а в ліву $y_1, \dots, y_n$

```
\newcommand{\nvec}[2][n]%
{\ensuremath{ #2_1,\dots,#2_{#1}}}
Візьмемо у праву руку
\nvec[k]{x}, а в ліву \nvec{y}
```

[25] Спробуйте скористатися командою `\pl` в якій-небудь формулі.

На закінчення зауважимо, що правила хорошого тону вимагають, щоб автор виносив визначення команд в преамбулу документа, а не ховав їх у нетрях тексту.

13.5.2 Нові оточення

Інколи зручніше створити власне оточення замість визначення нової команди. Нове оточення визначається таким чином:

`\newenvironment{`*ім'я оточення*`}[`*кіл-сть аргументів*`]{`*команди перед текстом*`}{`*кінцеві команди*`}`

Кількість аргументів є необов'язкови аргументом. Компіляція LaTeXом такого оточення відбувається методом підстановки. Коли зустрічається команда `\begin{`*назва оточення*`}`, то спочатку обробляється матеріал, що поміщений в аргумент `{`*команди перед текстом*`}`. Потім опрацьовується текст всередині оточення. Матеріал, що поміщений в аргумент `{`*кінцевої команди*`}`, опрацьовується, коли зустрілася команда `\end{`*назва оточення*`}`. Відмітимо, частую нові оточення визначаються як вдосконалені уже існуючі оточення.

Сам пишу собі епіграфи і сам пишу твори.
М. Жванецький

```
\newenvironment{epigraph}
{\begin{flushright}\normalfont\slshape}
{\end{flushright}}
\begin{epigraph}
Сам пишу собі епіграфи і
сам пишу твори.\\
М.~Жванецький \end{epigraph}
```

Наступний приклад ілюструє створення оточення для описання команд.

назва команди

```
\newenvironment{command}{\par\small\%
addvspace{3.8ex}\vskip -\parskip \noindent%
\begin{tabular}{|l|}\hline}%
{\\\hline\end{tabular}\par\addvspace{3.8ex}%
\vskip -\parskip}
\begin{command}
\em назва команди
\end{command}
```

Можливо, це досить складний приклад. В цьому прикладі текст оточення

розташовується у вигляді таблиці, що складається всього лиш із одного вічка. При цьому робиться верхній та нижній відступи для виділення тексту.

Іноді виникає потреба нумерувати елементи оточення. Для цього використовують лічильники (див. розділ 17 на стор. 80). Наведемо приклад оточення для створення білетів.

```
\newenvironment{bilet} {\begin{center}
Харківський національний університет імені В.Н. Каразіна\\
Навчальна дисципліна: Алгебра та геометрія.\\
\par Екзаменаційний білет \No \theNomerBileta \addtocounter{NomerBileta}{1}
\end{center}\par}{\par \bigskip Затверджено на засіданні кафедри
геометрії 17 грудня 2009 року  \par \bigskip
Екзаменатор \hskip 5 cm Д.І.~Власенко}
```

Далі задаємо початкове значення лічильника і створюємо білети.

```
\newcounter{NomerBileta}\setcounter{NomerBileta}{1}
\begin{bilet}
\item Перше питання.
\item Друге питання.
\end{bilet}
```

Харківський національний університет імені В.Н. Каразіна
Навчальна дисципліна: Алгебра та геометрія.
Екзаменаційний білет №1

1. Перше питання.
2. Друге питання.

Затверджено на засіданні кафедри геометрії 17 грудня 2009 року

Екзаменатор Д.І. Власенко

Зазначимо, що команда `\renewenvironment` перевизначає існуюче оточення.

13.5.3 Особисті пакети

Працюючи з LaTeX 2_ε Ви будете створювати нові команди та оточення, і в певний момент преамбула документа стане громіздкою. В цій ситуації розумно перемістити все своє «багатство» в окремий файл. Це нагадує перенесення

частин документа в окремі файли та підключення їх за допомогою команди `\input`. Нагадаємо, що за допомогою команди `\input` можна підключити до документу фрагмент тексту розташований у окремому файлі.

Настройки варто зберігати у файлі з розширенням `.sty`. При цьому на початку файла записується команда

`\ProvidesPackage{`*назва пакета*`}`

Тогді LaTeX 2_ε може правильно повідомити про помилки[26]. В документі файл настройк підключається в преамбулі командою `\usepackage`.

[26] Тьфу-тьфу-тьфу

Частина III
Набір формул

Немає межі досконалості.

Почнемо з невеличкої історичної розвідки. Створений Д. Кнутом TEX звичайно ж, надавав добрі можливості для набору математичних формул. Проте, за підтримки американського математичного товариства ($\mathcal{AMS}$), Майкл Співак розробив розширення TEXa, відоме як $\mathcal{AMS}$-TEX, яке дозволяло створювати математичні тексти будь-якої складності. В той же час Леслі Лампорт у розширенні LATEX запрограммував нові можливості для створення структурованних документів. Ці розробки не були, природно, сумісними. Переваги кожного з них втілилися в $\mathcal{AMS}$-LATEX, а саме в пакеті amsmath і пакеті символів amssymb, які підключаються в LATEX 2_ε стандартним чином.

14 Елементарна математика

В документі формули можуть розташовуватися всередині текста і можуть бути винесені в окремий рядок — так звані виокремлені формули. Із наведеного нижче прикладу видні відміни між формулами, які набрані всередині текста та виокремленими формулами. Спочатку Кнут використовував для виокремлення математичних формул комбінацію символів $, що не дозволяло визначити: формула починається чи, навпаки, закінчилася. В LATEX 2_ε початок і кінець формули легко розрізнити, що, у випадку помилки, призводить до менших втрат. Нижче наведені альтернативи для набору формул:

Всередині:	`$...$`	`\(...\)`	`\begin{math}...\end{math}`
Виключна:	`$$...$$`	`\[...\]`	`\begin{displaymath}...\end{displaymath}`

Можливо створювати виокремлені формули за допомогою оточення `equation`, тоді формули автоматично нумеруються. (Подробиці на стор. 73.)

Змінні a, b, c та α в рядку.

$$f(\alpha,\beta)=\alpha+\beta$$

```
Змінні \( a,b, c\) та $\alpha$ в рядку.
\[f(\alpha,\beta)=\alpha+\beta\]
```

Як видно із прикладу, TEX проігнорував пробіл після коми та знака долара. Пробіли[27] в формулах завжди ігноруються — TEX сам знає якак правильно ставити пробіли. TEX надає можливість вставляти горизонтальні пробіли. Ніколи не варто забувати про пробіли, що вказують на закінчення команды. Ще одне правило LATEX: порожні рядки в формулах заборонені — кожна формула займає лише один абзац.

Математична формула набирається шрифтом, що відрізняється від основного шрифта. Тому нвіть одну змінну потрібно набирати в математичному режимі.

14.1 Індекси: Верх і низ

Верхній та нижній індекси набираються за допомогою знаків `^`, `_` відповідно. Якщо індекс складається більше ніж із одного символа, то його потрібно брати в групові дужки `{...}`. Прояснимо можливі ситуації прикладами.

Квадрат довжини вектора (a_1, a_{222}) дорівнює $a_1^2+a_{222}^2$. Зауважте, TEX може ставити індекси так ${a_1}^2$ або так ${a_1}^2{}_3^4$.
А от степені:

$$a^{nm}, a^{nm}, a^{n^m}$$

```
Квадрат довжини вектора $(a_1,a_{222})$
дорівнює $a^2_1+a_{222}^2$. Зауважте,
\TeX{} може ставити індекси так
${a_1}^2$ або так \(a_1{}^2{}_3{}^4\).\\
А от степені: \[a^{nm},{a^n}^m, a^{n^m}\]
```

14.2 Дріб

Всередині текста дріб можна набирати за допомогою знака ділення «/». Для набору чисельника над знаменником потрібно використовувати команду

`\frac{`*чисельник*`}{`*знаменник*`}`

Якщо чисельник або знаменник дробу є лише один символ, то його можна не брати в групові дужки.

[27] Мається на увазі символ пробілу ␣. Все ж команди пробілів у математичн формулах, типа \␣, працюють, як і повинні.

$1/2+3/4$ дорівнює $\frac12+\frac 3 4$?\\
Косинус кута $$\frac{(a,b)}{|a||b|}$$

$1/2+3/4$ дорівнює $\frac12+\frac 3 4$?
Косинус кута
$$\frac{(a,b)}{|a||b|}$$

14.3 Дужки

Дужки можуть бути круглими, квадратними `[ ]`, вертикальними `| |`, фігурними `{ }` та кутовими $\langle\ \rangle$. Круглі, квадратні і вертикальні набираються безпосередньо, а фігурні і кутові відповідно командами `\{`, `\}`, `\langle` та `\rangle`. Також для набору квадратних і вертикальних дужок є дублюючі команди `\lbrack`, `\rbrack` и `\vert`.

$$\langle a,b\rangle = a_1b_1+\ldots+a_nb_n; \qquad (\frac12)^2 x\in[a;b]$$

```
$$\langle a,b \rangle = a_1b_1+
\ldots+a_nb_n;\qquad (\frac 12)^2
x\in[a;b\rbrack$$
```

У прикладах наведених вище розмір дужок фіксований. LaTeX 2_ε може сам подібрати розмір дужок. Для цього використовують команди

`\left`*дужка* ... `\right`*дужка*

Ці команди зажди потрібно використовувати парами. В протилежному випадку TeX повідомить про помилку. Коли одна із дужок зайва, тоді вона замінюється символом точки.

$$\left(\frac12\right)^2 \qquad \left.\frac{\partial f}{\partial x}\right|_{x=0} \qquad \left||x|+|y|\right|$$

```
$$\left(\frac 12\right)^2 \qquad
\left. \frac{\partial f}{\partial x}
\right|_{x=0} \qquad
\left||x|+|y|\right|$$
```

В останньому прикладі ставить скобки дужки потрібно ставити примусово. Для цього є набори дужок фіксованих розмірів:

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

$$\big||x|+|y|\big|$$ $\big||x|+|y|\big|$

Якщо виникне помилка з дужками, то TEX повідомить, що відсутня ліва або лишня права дужка командою `! Extra \right.`. Аналогічно текст помилки для іншой дужки — `! Missing \right. inserted.`

➢**Вправа 14.1**

Наберіть наступний текст. „Відкритий інтервал інколи позначають так: $x \in]a;b[$."

14.4 Матриці

У попередньому розділі ми навчились створювати дужки довільного розміру. Тепер, якщо взяти в дужки таблицю (див. п. 12.4), то отримаємо матрицю. Згадаємо, що оточення `tabular` використовується в текстовому режимі, а дужки ми набираємо в математичному. Тому для створення матриць слід використовувати оточення `array`, в якому елементи рядка розділяються символом `&`, а команда `\\` починає новий рядок. Для вирівнювання стовпчиків використовуємо обов'язковий параметр `l` (left), `r` (right) або `c` (center).

$$\begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}$$

$$\begin{cases} x = & a+b+2 \\ y = & b \end{cases}$$

```
$$ \left(\begin{array}{cc}
  1 & 2 \\
  1 & 1 \\
\end{array} \right)
\left\{\begin{array}{cl}
  x= & a+b+2 \\
  y= & b \\
\end{array} \right.  $$
```

Зверніть увагу, що фигурна дужка набрана як команда `\{`, звичайну фігурну дужку `{` використовують для початку групи.

14.5 Корінці і ... привиди

Знак кореня одержати вельми просто:

`\sqrt[`*показчик кореня*`]{`*підкореневий вираз*`}`

$$\sqrt[4n]{x^{4n}}=|x|,\ \sqrt{\sqrt{a}\sqrt{b}\sqrt{g}}$$

```
\[ \sqrt[4n]{x^{4n}}=|x|,\
\sqrt{\sqrt a\sqrt b\sqrt g} \]
```

Неозброєним оком видно, що в останньому прикладі корінь «зазвилювався». Він почав підстроюватися під висоту та глибину символів. Як вказувати TEXу висоту та глибину блока обговорювалось у розділі 12.5. Проте, це не наші методи — це ж не текстовий режим. Потрібно закликати смоків, вовкулаків, дідьків і решту. А називати їх будемо фантомами.

Перший із фантомів — страта. Тільки не та страта, що страта, а та страта що латинськими бувами пишеться як strut, анклійською мовою означає "розпірка" а у нас це `\mathstrut`. По вертикалі вона дещо вища b і дещо глибша ніж g. А ширина при цьому нульова у всіх одиницях вимірювання. Загалом, типовий привид звичайної круглої дужки. Випробуємо заклинання:

$$\sqrt{\sqrt{a}\sqrt{b}\sqrt{g}}$$

```
$$\sqrt{\sqrt{a\mathstrut}
\sqrt{b\mathstrut}\sqrt{g\mathstrut}}$$
```

— Ну, хто в наш час боїться привида дужки? — скаже читач, і буде правий. Ось він, головний головнокомандувач, вертикальный демон:

`\vphantom{`*math*`}`

який досягає вертикальних меж загиблої формули {*math*}.

— Однієї влади над вертикаллю недосить! Горизонталь потрібна! — каже вдумливий, уважний читач чи просто зануда. И одержить владу над усіма привидами формул:

`\phantom{`*math*`}`

$$\sqrt{\sqrt{}=\sqrt{}+\sqrt{}}$$

```
$$\sqrt{\sqrt{\phantom{(x+y)}}=
\sqrt{\phantom{b}}+\sqrt{\phantom{g}}}$$
```

Для створення големів використовують заклинання

`\smash[t/b]{`*math*`}`

Друкується справжній текст цієї формули, але TEX вважає висоту та глибину

формули такою, що дорівнює нулю. Пакет amsmath дозволяє використовувати необов'язковий аргумент: параметр `t` дозволяє нехтувати тільки висотою, а `b` — тільки глибиною.

14.6 Функції

TEX прожив уже багато років, тому багато функцій знає по імені. Якщо просто написати `$sin x$`, то TEX подумає, що це примхливо згруповані змінні, і на друк виведе «$sinx$». А коли сказати те ж саме командирським голосом `$\sin x$`, то TEX зрозуміє, що це — $\sin x$.

Назвемо поіменно функції:

arccos	`\arccos`	csc	`\csc`	ker	`\ker`	min	`\min`
arcsin	`\arcsin`	deg	`\deg`	lim	`\lim`	Pr	`\Pr`
arctan	`\arctan`	det	`\det`	lim inf	`\liminf`	sec	`\sec`
arg	`\arg`	dim	`\dim`	lim sup	`\limsup`	sin	`\sin`
cos	`\cos`	exp	`\exp`	lg	`\lg`	sinh	`\sinh`
cosh	`\cosh`	gcd	`\gcd`	ln	`\ln`	sup	`\sup`
cot	`\cot`	hom	`\hom`	log	`\log`	tan	`\tan`
coth	`\coth`	inf	`\inf`	max	`\max`	tanh	`\tanh`

Зауважимо, що в англомовній літературі деякі вітчизняні функции, такі як `\tg`, `\ctg` називаються незвично — `\tan`, `\cot`. Якщо такі функції ще не визначені, то їх можна визначити самостійно:

```
\newcommand{\tg}{\mathop{\rm tg}\nolimits}
```

В цьому визначенні команда `\nolimits` вказує, що індекси не можуть позиціонуватися над і під функцією, на відміну, скажемо, від $\sup_{y\in Y}$.

14.7 Інтеграли, суми, ...

Для одержання різних інтегралів використовують команди:

$\int$	`\int`	$\oint$	`\oint`	$\iint$	`\iint`
$\iiint$	`\iiint`	$\iiiint$	`\iiiint`	$\int\cdots\int$	`\idotsint`

Для вказівки меж інтегрування використовують верхній та нижній індекси. Межі інтегрування мають вигляд індексів, при необхідності їх можна підправити:

Зверніть увагу на тонкий пробіл:
$\int_a^{f(b)} f(x)\,dx \qquad \int\limits_a^b f(x)\,dx$

```
Зверніть увагу на тонкий пробіл:\\
$\int_a^{f(b)} f(x)\,dx \qquad
\int\limits_a^b f(x)\,dx$
```

Уважний читач уже напевно звернув увагу на команди

`\limits`	`\nolimits`

які дозволяють керувати роташуванням «меж». Перша команда розташовує їх внизу/вгорі операторів, друга ж вказує, що межі повинні бути розташовані як індекси.

Оператор суми «$\sum$» набирається як `\sum`. Межі підсумовування в абзаці і у виокремленій формулі TEX набирає по різному:

Степеневий ряд $\sum_{n=0}^\infty x^n$. Гармонічний —
$$\sum_{n=0}^\infty \frac 1n$$

```
Степеневий ряд $\sum_{n=0}^\infty x^n$.
Гармонічний~---
\[\sum_{n=0}^\infty \frac 1n\]
```

Для оператора суми і операторів, що наведені нижче, TEX розташовує межі як індекси, коли формула в абзаці і знизу/зверху оператора у виокремленій формулі.

$\sum$	`\sum`	$\bigcap$	`\bigcap`	$\bigsqcup$	`\bigsqcup`	$\bigodot$	`\bigodot`
$\prod$	`\prod`	$\bigcup$	`\bigcup`	$\bigvee$	`\bigvee`	$\bigotimes$	`\bigotimes`
$\coprod$	`\coprod`	$\biguplus$	`\biguplus`	$\bigwedge$	`\bigwedge`	$\bigoplus$	`\bigoplus`

Наведені вище оператори і інтеграли відносяться до «великих операторів» (тип `Op`). Вони зустрічаються на початку формули або підформули і

звичайно мають індекси. «Великі оператори», як правило, присутні у двох вимірностях — для текстового та виокремленого стилів. Важливо знати, що вони відрізняються від бінарних операторів (тип `bin`), таких як `\cap, \cup`. Бинарні ж операції зустрічаються між символами і лише зрідка мають індекси. І в жодному разі не переплутайте знак `\sum` і букву `\Sigma` (тип `alpha`).

14.8 Математика в таблицях

Деякі математичні конструкції:

f'	`\f'`	$\frac{abc}{xyz}$	`\frac{abc}{xyz}`
$\sqrt{abc}$	`\sqrt{abc}`	$\sqrt[n]{abc}$	`\sqrt[n]{abc}`
$\overrightarrow{abc}$	`\overrightarrow{abc}`	$\overleftarrow{abc}$	`\overleftarrow{abc}`
$\overline{abc}$	`\overline{abc}`	$\underline{abc}$	`\underline{abc}`
$\overbrace{abc}$	`\overbrace{abc}`	$\underbrace{abc}$	`\underbrace{abc}`
$\widetilde{abc}$	`\widetilde{abc}`	$\widehat{abc}$	`\widehat{abc}`

Букви грецької абетки:

α `\alpha`	κ `\kappa`	ς `\varsigma`	Λ `\Lambda`
β `\beta`	λ `\lambda`	τ `\tau`	Ξ `\Xi`
γ `\gamma`	μ `\mu`	υ `\upsilon`	Π `\Pi`
δ `\delta`	ν `\nu`	ϕ `\phi`	Σ `\Sigma`
ϵ `\epsilon`	ξ `\xi`	φ `\varphi`	Υ `\Upsilon`
ε `\varepsilon`	o `o`	χ `\chi`	Φ `\Phi`
ζ `\zeta`	π `\pi`	ψ `\psi`	Ψ `\Psi`
η `\eta`	ϖ `\varpi`	ω `\omega`	Ω `\Omega`
θ `\theta`	ρ `\rho`	Γ `\Gamma`	
ϑ `\vartheta`	ϱ `\varrho`	Δ `\Delta`	
ι `\iota`	σ `\sigma`	Θ `\Theta`	

Символи бінарних операцій:

$\pm$ `\pm`	$\times$ `\times`	$*$ `\ast`
$\mp$ `\mp`	$\div$ `\div`	$\star$ `\star`

$\circ$ \circ	$\wedge$ \wedge	$\ominus$ \ominus
$\bullet$ \bullet	$\setminus$ \setminus	$\otimes$ \otimes
$\cdot$ \cdot	$\wr$ \wr	$\oslash$ \oslash
$\cap$ \cap	$\diamond$ \diamond	$\odot$ \odot
$\cup$ \cup	$\bigtriangleup$ \bigtriangleup	$\bigcirc$ \bigcirc
$\uplus$ \uplus	$\bigtriangledown$ \bigtriangledown	$\dagger$ \dagger
$\sqcap$ \sqcap	$\triangleleft$ \triangleleft	$\ddagger$ \ddagger
$\sqcup$ \sqcup	$\triangleright$ \triangleright	$\amalg$ \amalg
$\vee$ \vee	$\oplus$ \oplus	

Символи відношень:

$\leq$ \leq \le	$\subseteq$ \subseteq	$\approx$ \approx
$\geq$ \geq \ge	$\supseteq$ \supseteq	$\cong$ \cong
$\leqslant$ \leqslant	$\sqsubseteq$ \sqsubseteq	$\neq$ \neq \ne
$\geqslant$ \geqslant	$\sqsupseteq$ \sqsupseteq	$\doteq$ \doteq
$\prec$ \prec	$\in$ \in	$\models$ \models
$\succ$ \succ	$\ni$ \ni \owns	$\perp$ \perp
$\preceq$ \preceq	$\vdash$ \vdash	$\mid$ \mid
$\succeq$ \succeq	$\dashv$ \dashv	$\parallel$ \parallel
$\ll$ \ll	$\equiv$ \equiv	$\smile$ \smile
$\gg$ \gg	$\sim$ \sim	$\frown$ \frown
$\subset$ \subset	$\simeq$ \simeq	$\propto$ \propto
$\supset$ \supset	$\asymp$ \asymp	$\bowtie$ \bowtie

Стрілочки:

$\leftarrow$	\leftarrow	$\Rightarrow$	\Rightarrow
$\rightarrow$	\rightarrow	$\Leftrightarrow$	\Leftrightarrow
$\leftrightarrow$	\leftrightarrow	$\mapsto$	\mapsto
$\Leftarrow$	\Leftarrow	$\leftharpoonup$	\leftharpoonup

$\leftharpoondown$	\leftharpoondown	$\longrightarrow$	\longrightarrow
$\rightleftharpoons$	\rightleftharpoons	$\longleftarrow$	\longleftarrow
$\hookleftarrow$	\hookleftarrow	$\longleftrightarrow$	\longleftrightarrow
$\hookrightarrow$	\hookrightarrow	$\longmapsto$	\longmapsto
$\rightharpoonup$	\rightharpoonup	$\Longrightarrow$	\Longrightarrow
$\rightharpoondown$	\rightharpoondown	$\Longleftarrow$	\Longleftarrow
$\uparrow$	\uparrow	$\Longleftrightarrow$	\Longleftrightarrow
$\Uparrow$	\Uparrow	$\nearrow$	\nearrow
$\downarrow$	\downarrow	$\searrow$	\searrow
$\Downarrow$	\Downarrow	$\nwarrow$	\nwarrow
$\updownarrow$	\updownarrow	$\swarrow$	\swarrow
$\Updownarrow$	\Updownarrow		

Різнорідні символи:

$\ldots$	\ldots	$\Re$	\Re	$\clubsuit$	\clubsuit
$\cdots$	\cdots	$\Im$	\Im	$\diamondsuit$	\diamondsuit
$\vdots$	\vdots	$\aleph$	\aleph	$\heartsuit$	\heartsuit
$\ddots$	\ddots	$\hbar$	\hbar	$\spadesuit$	\spadesuit
∞	\infty	∇	\nabla	$\dagger$	\dag
∂	\partial	$\triangle$	\triangle	$\ddagger$	\ddag
$\prime$	\prime	$\neg$	\neg	$\S$	\S
$\emptyset$	\emptyset	$\surd$	\surd	$\P$	\P
$\angle$	\angle	$\top$	\top	$\copyright$	\copyright
$\forall$	\forall	$\bot$	\bot	$\pounds$	\pounds
$\exists$	\exists	$\backslash$	\backslash	$\checkmark$	\checkmark
$\imath$	\imath	$\flat$	\flat	$\maltese$	\maltese
$\jmath$	\jmath	$\natural$	\natural	$\circledR$	\circledR
ℓ	\ell	$\sharp$	\sharp	$\yen$	\yen
$\wp$	\wp	$\|$	|	$\ulcorner$	\ulcorner

⌝	\urcorner	◇	\diamond	·	\cdot
⌟	\lrcorner	□	\Box		
⌞	\llcorner	℧	\mho		

15 Математичні типажі

Кожен математичний символ в TEXi відноситься до єдиного типу. При формуванні формули TEX, знаючи оператора, застосовує закладені в ньому правила. В TEXi візначені такі типи:

Тип	*Значення*	*Приклад*	*Ще приклад*
\mathalpha	Алфавітно-цифровой символ	F	\Sigma $= \Sigma$
\mathopen	Відкривна дужка	(	\langle $= \langle$
\mathclose	Закривна дужка	)	\rbrack $=]$
\mathbin	Бінарний оператор	\cup $= \cup$	\pm $= \pm$
\mathop	«Великий» оператор	\sum $= \sum$	\oint $= \oint$
\mathord	Звичайний математичний знак	\forall $= \forall$	\emptyset $= \emptyset$
\mathrel	Знак відношення	:	\leqslant $= \leqslant$
\mathpunct	Розділовий знак	;	\colon $= :$

15.1 Пробільні проблеми

Наведемо приклади, що показують, як в залежності від типу оператора, TEX оточує їх пробілами різної ширини. Почнемо із знака «:». Для нього в математичній моді є два випадки: розділовий знак (\colon) і відношення ($a : b$). Посля знака пунктуації TEX розташовує тонкий пробіл, а знак відношення виділяється пробілами з обох сторін.

$f|: A \to B$
$f| : A \to B$

```
$f|\colon A \to B$\\
$f|: A \to B$
```

Зверніть увагу, що пунктуація в текстовому і математичному режимі (ще кажуть моді) розрізняються. В текстовому режимі після знака пунктуації ставиться пробіл, а в математичному — тонкий пробел. Тому потрібно правильно використовуати знаки пунктуації:

Візьмемо a, b чи c. Це правильно.
Якщо a, b або c? Це не правильно.

```
Візьмемо~$a$, $b$ чи~$c$. Це правильно.\\
Якщо $a, b$ або $c$? Це не правильно.
```

Зверніть увагу, як з допомогою нерозривного пробілу знеможливлюється початок нового рядка із змінної a або c.

При правильному додержанні правил пунктуації рядки будуть верстаться правильно. В наведеному вище неправильному прикладі, TEX ніколи не розірве рядок між комою та `b`.

Крапка сприймається TEXом як звичайний символ. Тому десяткові дроби (3,14) можна записувати через крапку, а для запису за допомогою коми потрібно писати так — `$3{,}14$`.

Перед і після формули TEX вставляє додаткові пробіли. Їх величина визначається змінною `\mathsurround`. В plain TEX величина `\mathsurround=0pt`.

15.2 Шрифти у формулах

В математичній моді моді текст за недомовленістю набирається курсивом. А що робити, коли потрібно, наприклад, набрати щось подібне до X_{singular}? Або потрібно надрукувати вектор $\mathbf{a}$? Відповідь одна. Потрібно перейти до іншого шрифта. Для різних шрифтів є відповідні команди:

Шрифт	*Команда*	*Приклад*
Основний	`\mathnormal`	$x + T_y$
Прямий	`\mathrm`	$\mathrm{x + T_y}$
Напівжирний	`\mathbf`	$\mathbf{x + T_y}$
Рубаний	`\mathsf`	$\mathsf{x + T_y}$
Машинописний	`\mathtt`	$\mathtt{x + T_y}$
Каліграфічний	`\mathcal`	$\mathcal{X} + \mathcal{T}_{\mathcal{Y}}$
Курсивний	`\mathit`	$\mathit{x + T_y}$
Ажурний	`\mathbb`	$\mathbb{X} + \mathbb{T}_{\mathbb{Y}}$
Готичний	`\mathfrak`	$\mathfrak{x} + \mathfrak{T}_{\mathfrak{y}}$

Команди зміни шрифтів можна записувати без фігурних дужок, тоді аргументом буде слугувати лише перший символ, що йде за командою.

$\mathbb{Z}+\mathbb{R}$, $\mathbf{a}+b$

```
$$\mathbb{Z+R},\qquad \mathbf a+b$$
```

Інколи трапляється, що цих команд недосить, можливо варто скористатися командами:

`\boldsymbol`	для одержання напівжирних чисел та інших символів, що не є буквами, а також напівжирних букв грецької абетки
`\pmb`	для математичних символів, що не мають напівжирних варіантів
`\text`	для звичайного тексту. Пробіли ставляться як у звичайному текстовому режимі.

Для набирання тексту всередині формули можна використовувати команду `\mbox`, яка, як відомо, формує блок. Зокрема, при створенні блока можна використовувати математичні формули. Для одержання рамки навколо формули використовують команду `\boxed`.

Похилі букви грецької абетки одержують за допомогою префикса `\var`.

$\boxed{\varSigma+\varPsi}$, індекс$^{\textit{верхній}}_{\textbf{нижній}}$

```
$$\boxed{\varSigma+\varPsi},\quad
\text{індекс}^\text{\it верхній}_%
\text{\textbf{нижній}}$$
```

15.3 Розставимо крапки в крапках

Три крапки майже завжди набираються як `\dots`. Розташування крапок (по базовій лінії чи по центру) узгоджується із наступним символом. Якщо наступний символ — символ бинарної операції чи відношення, то крапки центруються, а коли символ пунктуації, то лягають на базу. Проте, наступним символом може бути закриття формули. От тоді TEX розгубиться. Турботливе $\mathcal{AMS}$ пропонує декілька команд, що розставляє всі крапки над крапками.

`\dotsc`	для крапок після коми
`\dotsb`	для крапок після бінарної операції чи відношення
`\dotsm`	для крапок замість множення
`\dotsi`	для крапок замість інтеграла

$a+b+\cdots\leqslant 3$ $A+B+\cdots$ $I_1I_2\cdots$

```
\[ a+b+ \dots \leqslant 3 \qquad
A+B+\dotsb\qquad I_1I_2\dotsm \]
```

15.4 Розтягувані стрілки

$$\text{Із пункта } A\xrightarrow[\text{запізнілий}]{\text{їхав потяг у}}\text{Л}$$

```
$$\text{Із пункта } A\xrightarrow[\text
{запізнілий}]{\text{їхав потяг у}}
\text{Л} $$
```

15.5 Символ зліва, символ справа...

Команда `\stackrel{`*символ*`}{`*бінарне відношення*`}` розташовує символ над бінарним відношенням. Більш загальні команди

`\overset` / `\underset{`*символ*`}{`*нерухомий символ*`}`

розташовує символ над/під символом.

$$f(x)\stackrel{def}{=}const \qquad \underset{1}{\overset{\infty}{\Omega}}$$

```
\[ f(x)\stackrel{def}{=}const \qquad
\underset{1}{\overset{\infty}\Omega} \]
```

Можна розташовувати символи як індекси зліва і справа від великого оператора. Для цього призначена команда

`\sideset{`*ліві індекси*`}{`*праві індекси*`}`

$$\leftarrow {}_3^2\bigoplus_4^1\uparrow\downarrow\rightarrow \qquad {}_*\sum_i{}' \lambda_i$$

```
\[ \leftarrow\sideset{_3^2}{^1_4}
\bigoplus^\uparrow_\downarrow\rightarrow
\qquad \sideset{_*}{'}\sum_{i}\lambda_i \]
```

15.6 Стилі математики

TEX керує размірами створюваних об'єктів за допомогою стилів. Всього використовується вісім стилів:

D, *D'*	`\displaystyle`	для виокремлених формул	text size
T, *T'*	`\textstyle`	для формул в тексті	text size
S, *S'*	`\scriptstyle`	для верхніх чи нижніх індексів	script size
SS, *SS'*	`\scriptscriptstyle`	для індексів до індексів	scriptscript size

Стилі з акцентами — це стислі стилі (cramped). У них индекси сильніше стискають по вертикалі. TEX звертається до цих стилів коли має справу з

індексами та дробами. В якому випадку TEX надає перевагу тому чи іншому стилю описано в в таблиці:

Стиль	верхній індекс	нижній індекс	чисель-ник	знамен-ник
D	S	S'	T	T'
D'	S'	S'	T'	T'
T	S	S'	S	S'
T'	S'	S'	S'	S'
S, SS	SS	SS'	SS	SS'
S', SS'	SS'	SS'	SS'	SS'

Таким чином можна керувати зовнішнім виглядом формули. В наступному прикладі використовується команда `\dfrac=\displaystyle\frac`. Таке скорочення уже визначене в TEXi. Також визначена команда `\tfrac=\textstyle \frac`.

$$y_0+\frac{x_0}{y_1+\frac{x_1}{y_2+\frac{x_2}{y_3+\frac{x_3}{y_4}}}},\quad y_0+\frac{x_0}{y_1+\frac{x_1}{y_2+\frac{x_2}{y_3+\frac{x_3}{y_4}}}}$$

```
\[ y_0+\frac{x_0}{y_1+\frac{x_1}{y_2+
\frac{x_2}{y_3+\frac{x_3}{y_4}}}},\quad
y_0+\frac{x_0}{y_1+\dfrac{x_1}{y_2+
\dfrac{x_2}{y_3+\dfrac{x_3}{y_4}}}}  \]
```

15.7 Розриви у формулах та довгі формули

Розрив формули, що занурена в текст відбувається після знака бінарного відношення (rel) або після знака бінарного оператора (binop). Розрив у викоремленій формулі заборонений. Для набору багаторядкових виокремлених формул використовують спеціальні оточення.

Коли потрібно уникнути локального розриву, можна відповідну частину формули записати як групу, тобто взяти у фигурні дужки. Звертаємо увагу, що групування можна застосовувати для усунення небажаних пробілів, воно забороняє зміну міжсимвольнх інтервалів для вирівнювання рядків.

$1+2=2+1$
$1+2{=}2+1$
$1+2{=}2+1$
$1+2\mathrel{=}2+1$

```
${1+2}=2+1$\\
${1+2}{=2+1}$\\
$1+2{=}2+1$\\
$1+2\mathrel{=}2+1$
```

Для глобальної заборони розриву у формулах варто змінити параметри TEXa

```
\binoppenalty=10000
\relpenalty=10000.
```

Якщо використовувати ці формули для окремих частин документу, то слід знати, що розміри штрафів дорівнюють відповідно 700 и 500.

Для набору багаторядкових формул використовують різні оточення. Якщо вказати ім'я оточення із зірочкою наприкінці, то формули оточення не нумеруються. Розглянемо приклад оточення.

$$(a+b+c)^1 = a+b+c \quad (1)$$
$$(a+b)^2 = a^2+2ab+b^2 \quad (2)$$

```
\begin{align} (a+b+c)^1&=a+b+c\\
(a+b)^2&=a^2+2ab+b^2
\end{align}
```

Із прикладу видно, що для вирівнювання використовується символ &, а для разриву рядків — \\. Уважний читатч помітить, що це нагадує роботу із таблицями. Так воно і є. Фактично це є надбудова над таблицею. Оточення `split` використовується для розбивання формул та вирівнювання.

$$\begin{aligned}(a+b)^0 &= (a+b)^{1-1} \\ &= (a+b)^1 * (a+b)^{-1} \\ &= (a+b)^1/(a+b)^1 = 1\end{aligned} \quad (3)$$

```
\begin{equation}
\begin{split}
(a+b)^0 &=(a+b)^{1-1} \\
        &=(a+b)^1*(a+b)^{-1}\\
        &=(a+b)^1/(a+b)^1=1
\end{split}
\end{equation}
```

Довгі формули можна розбивати ще оточенням `\multline`, всередині якого використовуються команди `\shoveleft{...}` та `\shoveright{...}` для вирівнювання частини формули. За недомовленістю відбувається центрування.

Для набирання декількох формул без вирівнювання використовують оточення `\gather`. Приклад використання оточення є у наступному пункті.

15.8 Нумерація формул

Як уже казали, оточення `equation` автоматично породжує номерг номер виокремленої формули, а `equation*` створює виокремлену формулу без номера. При наборі виокремленої формули можна відмовитися від нумерації формули, або можна позначити її міткою за своїм вибором, також можливо не друкувати дужки, що оточують номер формули. Для таких випадків використовують команди

`\notag`	`\tag{`*метка*`}`	`\tag*{`*метка*`}`

$$1 = 1 \tag{4}$$
$$(a+b)^2 = a^2 + 2ab + b^2 \tag{меточка}$$
$$2 = 2 \tag{4′}$$

```
\begin{gather}
1=1 \label{eq:ex}  \\
(a+b)^2=a^2+2ab+b^2 \tag{меточка}\\
2=2 \tag*{\ref{eq:ex}$'$}
\end{gather}
```

Вигляд номерів формул можна пов'язати, наприклад, із нумерацією розділів, перевизначивши команду

```
\renewcommand{\theequation}{\arabic{equation}}
```

Але тоді нумерація формул буде наскрізною. Більш зручна команда, яка занулює лічильник на початку розділу:

`\numberwithin{`*equation*`} {`*subsection*`}`

Для посилань на формули передбачена команда `\eqref`, яка бере посилання на формулу в круглі дужки. Наприклад, такий вигляд має посилання на рівняння із останнього прикладу (4).

Розташування виокремлених формул можна налаштувати за допомогою опцій пакета amsmath. Номер формули можна розташовувати зліва — опція [*leqno*] або справа — [*reqno*] або з фіксованим відступом від лівого поля — опція [*fleqn*]. Коли остання опція не використовується, то за недомовленістю формула центрується. Відступ для опції [*fleqn*] дорівнює відступу виокремленої формули `\mathindent` (за недомовленістю `2.5em`).

15.9 Оточуємо теореми

При написанні статті Ви напевно повинні робити (оформляти) твердження. І, щоб вони не потонули в хаосі слів, їх потрібно виділяти і, до того ж, і мітити [28].

Визначаємо предивно гарне оточення-теорему:

Ттеорремма 1 (Оціночна) *Чим темніша ніч, тим яскравіші зорі.*

Зверніть увагу, що теорема 1 виділена вертикальними пробілами.

```
Визначаємо предивно гарне оточення-
теорему: \newtheorem{ttt}{Ттеорремма}
\begin{ttt}[Оціночна] \label{zub:1}
Чим темніша ніч, тим яскравіші зорі.
\end{ttt}
Зверніть увагу, що теорема \ref{zub:1}
виділена вертикальними пробілами.
```

В загальному вигляді команда створення нових оточень типу теорем має наступний вигляд:

```
\newtheorem{ім'я оточення}[лічильник]{текст заголовка}[розділ]
```

Якщо читач не квапився, то зміст обов'язкових параметрів йому уже зрозумілий. Лишається пояснити зміст необов'язкових. Параметр [*лічильник*] вказує яким із уже існуючих лічильників потрібно скористатися. Наприклад, у теоремі, лемі чи у твердженні, як правило, один лічильник на всіх. Далі ми користуємося лічильником, що визначений в предивно гарному прикладі. Параметр [*раздел*] визначає підпорядкованість лічильника розділу, в протилежному випадку йде наскрізна нумерація.

➢Вправа 15.1

Як часто зустрічаються команди `\newtheorem`, що мають два необов'язкові параметри?

Визначаємо нормальне, але підпорядковане оточення-теорему:

Теорема 2 *Чим далі в ліс, тим ближче виліз.*

Звертаємо увагу, що теорема має номер 2.

```
Визначаємо нормальне, але підпорядковане
оточення-теорему:
\begin{teo}\label{teo:1}
Чим далі в ліс, тим ближче виліз.
\end{teo}
Звертаємо увагу, що теорема
має номер~\ref{teo:1}.
```

[28] Вони ж бо маяки серед ночі хибних уявлень.

Пакет theorem дозволяє попрацювати над оформленням стилю тверджень. Ці команди потрібно розташовувати в преамбулі. Вибір стилю для оточення-теорем здійснюється командою

`\theoremstyle{`*им'я стилю*`}`

Існують наступні стилі теорем:

`plain`	Повторює стандартне визначення з поправкою на параметри `\theorempreskipamount` і `\theorempostskipamount`
`change`	Міняються місцями номер і назва заголовка
`margin`	Номер заголовка розташовується на лівому полі
`break`	Заголовок теореми розташовується на окремому рядку
`changebreak`	Міняються місцями номер і назва заголовка, які розташовуються на окремому рядку
`marginbreak`	Номер заголовка розташовується на лівому полі, а заголовок на окремому рядку

Вертикальні відступи задаються змінними `\theorempreskipamount` та `\theorempostskipamount`, які встановлюються командою `\setlength{`*команда*`}{`*довжина*`}`.

Вибір шрифта заголовка і формулювання виконується командами

`\theorembodyfont{`*шрифт*`}` `\theoremheaderfont{`*шрифт*`}`

Приклад експеримента:

3 Приклад (Спостереження) *Чим далі в ліс, тим товщі партизани.*

Одержали номер 3.

```
\theoremstyle{margin}
\theorembodyfont{\slshape}
\theoremheaderfont{\scshape}
\newtheorem{exa}[ttt]{Приклад}
Приклад експеримента:
\begin{exa}[Спостереження] \label{exa:1}
Чим далі в ліс, тим товщі партизани.
\end{exa}
Одержали  номер \ref{exa:1}.
```

Частина IV
Верхній пілотаж

. . . пташку жалко.

Щоб побудувати будинок потрібні цегла і розчин. З «цеглиночками» TEXa — блоками, ми уже познайомилися. Прийшла черга до клею, що з'єднує блоки. Проте, наш клей — дивний, він запросто може нескінченно розтягуватися чи розтягуватися пропорційно. Ще ми навчимося керувати лічильниками та відстанями, навчимося мати справу з графікою, а потім все повісимо в рамочку на стінку.

16 Клей

Ми вже казали про те, що TEX працює з боксами. А з'єднує він їх за допомогою чаклунського розчину, який будемо запросто називати клеєм. Можна до смішного просто довести, що клей існує. По-перше, якби TEX не міг виміряти відстань між блоками, то як би він міг вирівнювати рядки по ширині? А по-друге, якби його не було, то для чого говорити про нього? TEX заповнює клеєм пробіли між словами, але не між буквами — букви стоять впритул.

У клея є три атрибути — його природна величина (space), здатність розтягуватися (stretch) і здатність стискатися (shrink).

16.1 Горизонтальні проміжки

Горизонтальні проміжки вже обговорювались у підрозділі 10.8. Нагадаємо, що горизонтальний інтервал створюється командою `\hspace{`*довжина*`}`. Уточнимо, ця команда може вставляти найсправжнісінький клей, який і описується в опції {*довжина*}. В довжині необхідно вказати всі три атрибути клею. Наприклад, можлива команда `\hspace{2cm plus 5mm minus 1ex}`. В цій команді TEXу пропонується вставити клей шириною 2 сантиметри, максимально растяжний на 5 міліметрів, і який може стискатися не більше, ніж

на висоту однієї букви x. Зверніть увагу, що довжина клею може дорівнювати 0 cm.

А як вказати нескінченно розтяжний інтервал? Такий інтервал має довжину \hfill. Ось приклад трансляції коду \hfil \hfil \fbox{Дві третини на третину} \hfil:

Дві третини на третину

В цьому прикладі проміжок зліва відноситься до проміжку справа, як 2 : 1. Тепер читачу повинно бути зрозумілим, яким чином можна вирівнювати текст по центру. Зауважимо, що за визначенням клей \stretch{n}={n\fill}. При цьому значення параметра може бути десятковим дробом. Якщо потрібно щоб проміжок не зникав, коли він попадає на кінець рядка, потрібно використовувати \hspace*. Проміжок можливо заповнювати як горизонтальною прямою, так і послідовністю крапок —

\hrulefill \dotfill

зліва в центрі справа
зліва в центрі справа
зліва справа
зліва справа

```
\newcommand{\hsps}{\hspace{\stretch{1}}}%
зліва\hsps в центрі\hsps справа\\
зліва\hrulefill в центрі\hrulefill справа\\
зліва\hsps \hrulefill \hsps справа\\
зліва\dotfill \hrulefill\dotfill справа\\
```

16.2 Вертикальні проміжки

Вертикальний аналог команди \hspace — команда \vspace{*довжина*}. Варіація команди \vspace*{*довжина*} дозволяє створювати вертикальні проміжки, які не ігноруються. Нагадаємо, що нескінченною довжиною може бути \stretch{n}={n\fill}. В TEXi визначені стандартні величини вертикальних проміжків:

\smallskip	За недомовленістю дорівнює приблизно четвертій частині міжрядкового інтервала (\baselineskip) плюс чи мінус третина власної величини, тобто 1/12 (\baselineskip)
\medskip	Подвоєнний \smallskip
\bigskip	Подвоєний \medskip, тобто \baselineskip
\vfill	Нескінченно растяжний проміжок

Пропонується вказувати проміжки саме в цих величинах. Взагалі, при виборі клею для вертикального проміжка варто вказувати розтяжність та стискуваність в межах естетичного сприйняття документа.

➢**Вправа 16.1**

Використовуючи разтяжні пробіли створіть титульний лист будь якої книги.

16.3 Керування відстанню

Змінні, що є довжинами, створюються командою

`\newlength{`*ім'я команди довжини*`}`

Змінна довжини, що створюється, має нульове значення. Змінити значення змінної довжини можна командами

`\setlength{`*ім'я*`}{`*довжина*`}` `\addtolength{`*ім'я*`}{`*довжина*`}`

Перша команда створює величину довжини, а друга змінює її. Наведемо приклад, в якому варто звернути увагу на те, що друк величини довжини здійснюється за допомогою команди `\the`

Задана довжина 0.0pt
Задана довжина 3.0pt plus 1.0pt minus 2.84526pt
Задана довжина 6.0pt plus 2.0pt minus 5.69052pt

```
\newlength{\test}
Задана довжина \the\test\\
\setlength{\test}{ 3pt plus 1pt minus 1mm}
Задана довжина \the\test\\
\addtolength{\test}{ 3pt plus 1pt minus 1mm}
Задана довжина \the\test
```

Команда, яка дозволяє надати змінній значення ширини набраного текста —

`\settowidth{`*ім'я змінної*`}{`*текст*`}`

Довжина цього текста=111.59572pt

```
\settowidth{\test}{Довжина цього текста}
Довжина цього текста=\the\test
```

Подібним чином можна виміряти висоту та глибину текста за допомогою команд

`\settoheight{`*ім'я змінної*`}{`*текст*`}` `\settodepth{`*ім'я змінної*`}{`*текст*`}`

16.4 Макет смуги набору

Використовуючи команди зміни довжини можни редагувати макет смуги набору документа. Надрукувати макет документа можна командою из пакета layout

`\layout`

Макет цього документа і параметри, що керують макетом смуги наведені на рис. 5 на стор. 81. Опишемо декілька параметрів макета смуги.

Параметр `\oddsidemargin` додає вертикальний інтервал зліва на непарних смугах при двосторонньому друку, в протилежному випадку додається на всіх смугах. Його колега `\evensidemargin` додає інтервал на парних смугах тілько при двосторонньому друку.

При багатоколонковому наборі параметр `\columnsep` визначає ширину проміжка між колонками. Колонки можна відокремлювати також лінійкою. Товщина лінійки задається параметром `\columnseprule`. За недомовленістю цей параетр нульовий, тому лінійка невидима. Використовуючи ці параметри LaTeX разраховує ширину колонки — змінна `\columnwidth`.

17 Лічильники

При трансляції документа LaTeX нумерує значну кількість об'єктів, починаючи із глав та розділів і закінчуючи вкладеними нумерованими списками. Для цієї мети він використовує лічильники. Перерахуємо імена всіх лічильників, що використовуються в стандартних класах:

`part`	`paragraph`	`figure`	`enumi`
`chapter`	`subparagraph`	`table`	`enumii`
`section`	`page`	`footnote`	`enumiii`
`subsection`	`equation`	`mpfootnote`	`enumiv`
`subsubsection`			

Для оточення, що створюється командою `\newtheorem`, автоматично створюється одноіменний лічильник.

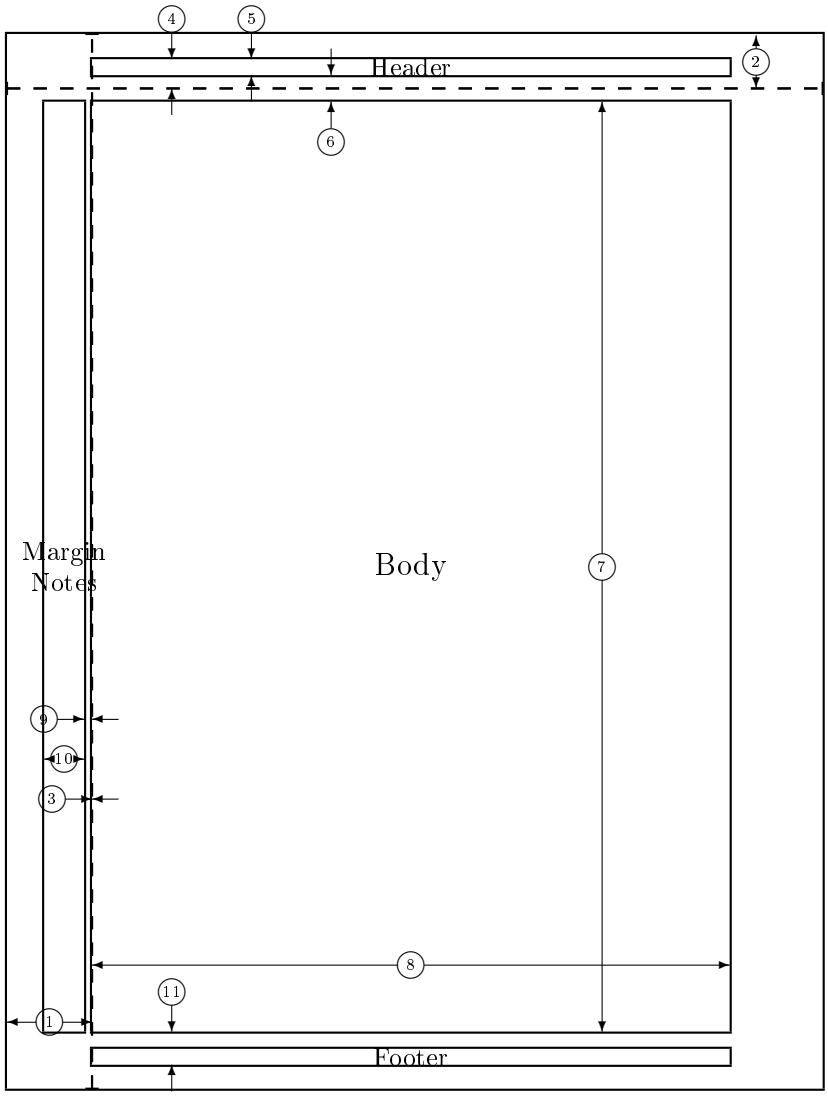
В LaTeXi можна створювати цілочисельні лічильники за допомогою команди

`\newcounter{`*ім'я нового лічильника*`}[`*ім'я старого лічильника*`]`

Ця команда створює новий лічильник із значенням нуль. Коли є необов'язковий параметр, вказується підпорядкування створюваного лічильника старому, ім'я якого вказано у необов'язковому параметрі. При підпорядкуванні лічильників зміна значення старшого лічильника командою `\stepcounter` чи `\refstepcounter` тягне за собою занулення усіх підпорядкованих лічильників.

Створення лічильника носить глобальный характер і LaTeX знає все про лічильник, навіть коли він був описаний у групі. Це відрізняє команду `\newcounter` від `\newcommand` та `\renewcommand`, які мають локальний характер.

Для зміни значення лічильника використовують команди призначення та




```
 1   one inch + \hoffset       2   one inch + \voffset
 3   \oddsidemargin = 0pt      4   \topmargin = -22pt
 5   \headheight = 12pt        6   \headsep = 20pt
 7   \textheight = 700pt       8   \textwidth = 480pt
 9   \marginparsep = 6pt      10   \marginparwidth = 30pt
11   \footskip = 25pt              \marginparpush = 0pt (not shown)
     \hoffset = -8pt               \voffset = -31pt
     \paperwidth = 614pt           \paperheight = 794pt
```

Рис. 5. Макет смуги набору. Тут наведені ті параметри, які використовуються в даному документі.

додавання

`\setcounter{`*ім'я лічильника*`}{`*величина*`}` `\addtocounter{`*ім'я лічильника*`}{`*величина*`}`

Значення лічильника добувається командою (як правило, для застосування в наведених вище командах)

`\value{`*ім'я лічильника*`}`

Збільшення значення лічильника та занулення всіх йому підпорядкованих лічильників здійснюється за допомогою

`\stepcounter{`*ім'я лічильника*`}`

Для створення посилань використовують команду, що схожа на попередню

`\refstepcounter{`*ім'я лічильника*`}`

Потрібно сказати, що при створенні лічильника автоматично визначається команда для друку його значения. Ця команда одержується із префикса `\the` та {*ім'я лічильника*}. Проілюструємо сказане.

Значення test=2002 чи
Значення test=MMII?

```
\newcounter{test}
\addtocounter{test}{2002}
Значение test=\thetest\ чи\\
Значение test=\Roman{test}?
```

Із приклада ми бачимо, що значення лічильника може бути надруковане римськими цифрами. В наведеній ничже таблиці перераховані все можливі написания значення лічильників.

`\arabic{`*лічильник*`}`	арабськими цифрами
`\roman{`*лічильник*`}`	римськими цифрами з використанням малих латинських букв
`\Roman{`*лічильник*`}`	римськими цифрами з використанням великих латинських букв
`\alph{`*лічильник*`}`	малими латинськими буквами — a, b,. . . , z. Значення лічильника не повинно перебільшувати 26
`\Alph{`*лічильник*`}`	великими латинськими буквами — A, B,. . . , Z. Значення лічильника не повинно перебільшувати 26
`\fnsymbol{`*лічильник*`}`	символами для маркування виносок в оточенні `minipage`. Використовується виключно у математичному режимі. Значення лічильника не повинно перебільшувати 9
`\the`*лічильник*	Команда виведення на друк значення лічильника у вигляді, що визначений за допомогою перерахованих вище команд

Перевизначимо друк введеного вище лічильника `\test` так, щоб при його друку виводився номер розділу і підрозділу, в якому виводиться значення цього лічильника.

Значение test=17.0.2003

```
\renewcommand{\thetest}
{\thesubsection.\arabic{test}}
\refstepcounter{test}\label{tst}
Значение test=\ref{tst}
```

➢Вправа 17.1

Визначте оточення для вправ із лічильником, що підпорядкований номеру розділу.

18 Графіка

Як ми пам'ятаємо, TEX-будівельник працює лише з боксами та клеєм. Його зовсім не хвилює вміст цих чорних ящищків. Зараз TEX, схоже, кращий із будівельників. У нас, звичайних користувачів, часто виникає бажання покласти в один із блоків який-небудь рисунок. Як це краще зробити ми і

обговоримо.

Перший спосіб полягає у тому, щоб включати зображення, створені в спеціальних графічних редакторах. Другий спосіб полягає в текстовому описанні рисунка за допомогою LaTeXa. Другий спосіб нагадує послідовне креслення на дошці елементів рисунка і використовується для створення простих рисунків.

Графічні зображення можна розташовувати в оточеннях `figure` и `table`, які підписують і нумерують свій вміст.

18.1 Імпортна графіка

Мабуть, імпортування — це найбільш простий спосіб використання графіки при тій невеличкій умові, що Ви маєте змогу створювати графіку в якомусь графічному редакторі. Створений вами рисунок принципіально може бути лише у двох форматах — або у векторному, або у растровому форматі. У більшості випадків є бажанішим векторний формат рисунка.

Із усіх можливих форматів LaTeX краще всього працює із графікою, що описана на мові PostScript. Расширення таких файлів — EPS (Encapsulated PostScript). Наприклад, в графічному редакторі Corel є можливість експортувати створений рисунок у EPS-формат.

Для включення рисунка в документ перш за все необхідно підключити пакет graphicx. Пакету потрібно вказати ім'я драйвера, за допомогою якого вставляється рисунок в dvi-файл. Це як правило `dvips`. Отже, якщо написати у преамбулі

```
\usepackage[dvips]{graphicx}
```

то це буде правильно.

Для включення файла графіки в документ використовується команда

```
\includegraphics[ключ=значення]{имя файла}
```

Можливе використання наступних ключів:

`width`	масштабує графіку до вказаної ширини
`height`	масштабує графіку до вказаної висоти
`angle`	обертає графіку проти годинникової стрілки на вказаний кут
`scale`	масштабує графіку

Другий варіант — включення рисунків за допомогою команди

`\includegraphics[`*0,0*`][`*ширина, висота*`]{`*ім'я файла*`}`

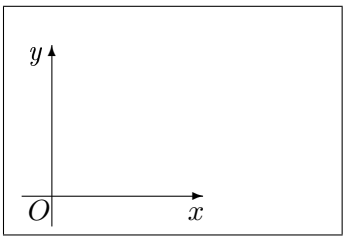
18.2 Своїми руками

В LaTeX передбачена можливість створення простих рисунків за допомогою оточня `picture`. Проте, набір команд і можливостей цього оточення обмежені, тому часто використовують пакети, що збільшують можливості `picture`.

Існує декілька спеціалізованих пакетів: fancybox (пункт 18.3) дозволяє створювати декоративні рамки, пакет bar креслить гістограми, бінарні та тернарні дерева простіше створювати пакетом trees. Є спеціальні пакети для креслення фейнмановських діаграм, зображення хімічних формул, електронних схем.

18.2.1 Стандартна `picture`

Розмову про графіку почнемо з прикладу.




```
\setlength{\unitlength}{.4cm}
\fbox{
\begin{picture}(10,7)
\put(0,1){\vector(1,0){6}}
\put(1,0){\vector(0,1){6}}
\put(0.2,0.2){$O$}
\put(0.2,5.5){\itshape y}
\put(5.5,0.2){$x$}
\end{picture} }
```

Перша команда встановлює довжину `\unitlength`, що буде одиницею вимірювання для рисунка. Отже вона є масштабною величиною, яка дозволяє

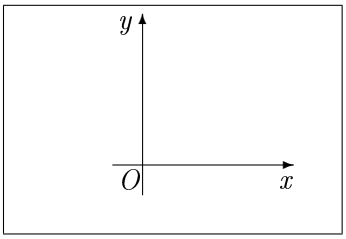
збільшувати чи зменшувати рисунок. Потім увесь рисунок береться в рамку знайомою командой `\fbox` для того, щоб показати розмір блока, який відведено під рисунок. Розмір цього блока задається в параметрі оточення `picture`: вказується ширина і висота. Зверніть увагу на круглі дужки! Оточення `picture` — єдине, параметри якого беруться у круглі дужки. Оточення `picture` дозволяє виконати зсув (паралельне перенесення) рисунка. Вектор зсуву вказується другим аргументом і також береться у круглі дужки. Наприклад, в «`\begin{picture}(10,7)(2,1)`» вказаний зсув на 2 одиниці вліво і на 1 вниз. Також вказано, що розмір рисунка — 10 одиниць по ширині і 7 по висоті.

Коли LaTeX знаходиться в оточенні `picture`, він знаходиться у системі координат, вісь Ox якої напрямлена вліво, а вісь Oy — вверх. При цьому координати об'єктів не обмежені розмірами рисунка.

Наступна важлива команда `\put` дозволяє розташовувати блоки на рисунку. Блоком может бути як текст, так і вектор чи інший рисунок. Таким чином, накреслена система координат може бути використана другий раз в іншому рисунку за допомогою команди `\put`. Команда `\put` розташовує блок так, щоб його лівий нижній ріг знаходився у вказаних координатах.

`\put(`*x,y-координати нижнього лівого рогу блока*`){`*блок*`}`

В наступному прикладі командою `\put` зсунемо створену систему координат на вектор $(3, 1)$:




```
\fbox{
\begin{picture}(10,7)
\put(3,1){
\put(0,1){\vector(1,0){6}}
\put(1,0){\vector(0,1){6}}
\put(0.2,0.2){\itshape O}
\put(0.2,5.5){\itshape y}
\put(5.5,0.2){\itshape x} }
\end{picture} }
```

Відрізок і вектор (стрілка) створюються командами

`\line(`x,y`){`*довжина більшої проекції*`}` `\vector(`x,y`){`*довжина більшої проекції*`}`

Параметр (x,y) вказує напрям лінії. Фактично цей параметр вказує вектор, в напрямі якого випускається лінія. Довжина цієї лінії вибирається TEX'ом так, щоб більша із проекцій на вісі Ox та Oy дорівнювала аргументу команди.

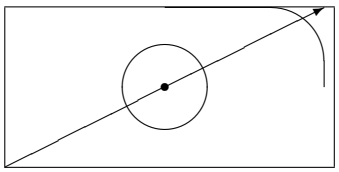
Проте, оскільки TEX складає лінію із невеличких відрізків, зображення яких зберігаються в окремих шрифтах, то існують обмеження на напрями, які можна задавати. По-перше, напрями задаються цілими числами. По-друге, числа, що задають напрями, повинні бути взаємно простими. По-третє, по модулю ці числа обмежені: для лінії вони не перевищують 6, для вектора — чотирьох.

Коло створюється командою `\circle{`*діаметр*`}`. Круг — командою `\circle*{`*діаметр*`}`. Часто ця команда використовується для позначення точок. В стандартному TEX'і діаметри кругів обмежені. TEX складає коло чи круг із шматочків своїх шрифтів. Тому, коли замовленого розміру немає, тоді береться найближчий із наявних. Із сказаного випливає, що максимальний діаметр кола обмежений. Щоб уникнути таких складних обмежень можна використовувати пакети типу curves. Цей пакет обговорюється в пункті 18.2.2 на стор. 90.

Овальні рамки — прямокутники с закругленними кутами, створюються командою

`\oval(`*ширина*`,`*висота*`)[`$t/b/r/l$`]`

Можна малювати половинки та четвертинки овальных рамок. Яку саме частину потрібно малювати вказується у необов'язковому аргументі. Наприклад, якщо вказано `tr` (top & right), то буде намальована верхня права чверть.

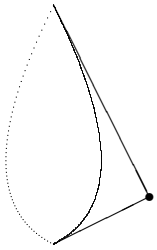


```
\begin{picture}(120,60)
\put(60,30){\circle{30}}
\put(60,30){\circle*{3}}
\put(60,30){\oval(120,60)[tr]}
\put(0,0){\vector(2,1){120}}
\end{picture}
```

Можна створювати квадратичні сплайни Без'є за допомогою команди

`\qbezier[`*кількість точок кривої*`](`x`,`y`)(`x`,`y`)(`x`,`y`)`

Якщо використовувати невелику кількість точок для зображення кривої, то крива може стати розрідженою. Але час на розрахунок кривої скоротиться. Максимальна кількість точок прямої зберігається у змінній `\qbeziermax`. Значення цієї змінною за недомовленістю дорівнює 500.

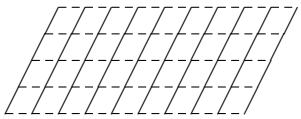


```
\begin{picture}(120,90)
\put(96,18){\circle*{3}}
\qbezier(60,0)(96,18)(60,90)
\qbezier[60](60,0)(24,18)(60,90)
\put(60,0){\line(2,1){36}}
\put(60,90){\line(1,-2){36}}
\end{picture}
```

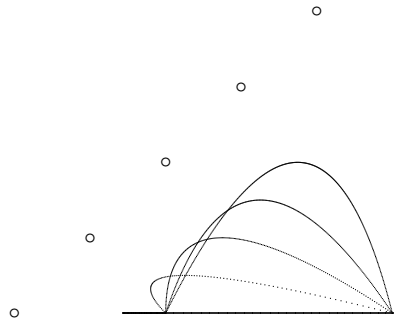
При розташуванні декількох однакових об'єктів команду `\put` можливо замінити однією командою. Для рівномірного розташувння декількох копій одного і того ж зображення використовують команду

`\multiput(`x`,`y`)(`Δx`,`Δy`){`*количество копий*`}{`*рисунок*`}`

Цю команду можна пристосувати для креслення пунктирних ліній або для крслення сітки. Хоча для цієї мети більш підходять команди із спеціалізованих пакетів. В першому прикладі створимо похилу сітку, а в другому серію кривих Без'е.



```
\begin{picture}(120,40)
\multiput(0,0)(10,0){11}{\line(1,2){20}}
\multiput(0,0)(5,10){5}{
\multiput(0,0)(5,0){20}{\line(1,0){3}} }
\end{picture}
```



```
\setlength{\unitlength}{.01cm}
\newcounter{N}\setcounter{N}{0}
\begin{picture}(600,500)
\multiput(0,0)(0,0){5}{
\qbezier[\value{N}](200,0)%
        (\value{N},\value{N})%
        (500,0)
\put(\value{N},\value{N}){\circle{10}}
\addtocounter{N}{100}  }
\end{picture}
```

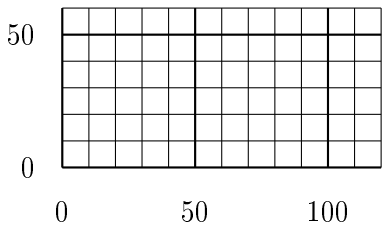
➢Вправа 18.1

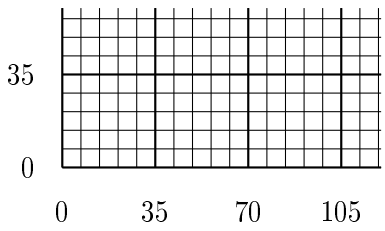
Придумайте приклад команди, що креслить сім'ю кривих, використовуючи один чи більше лічильників.

Пакет graphpap визначає команду

`\graphpaper{`*крок сітки*`}(`*лівий*`,`*нижній угол*`)(`*ширина*`,`*висота*`)`

яка створює пронумеровану координатну сітку. Нижче наведені дві сітки. Перша створена командою `\graphpaper(0,0)(120,60)`. За недомовленістю, необов'язковий аргумент кроку сітки дорівнює 10. У другому прикладі крок сітки дорівнює 7.





Розташуваня тексту в оточенні `picture` підкоряється правилам розташування блоків. Тому вільно можна використовувати команди формування блоків типу `\fbox`. Блоки описувались у розділі 12.5. Пізніше ми ще обговоримо роботу з блоками.

Лишається вказати, які параметри можна використовувати при створенні рисунків. Один ми вже знаємо. Це параметр масштабування `\unitlength`.

Зауважимо, що змінюючи величину `\unitlength` можна масштабувати окремі частини рисунка, не обов'язково весь.

Регулювати товщину ліній можливо за допомогою команд

`\thinlines` `\thicklines` `\linethickness{`*товщина*`}`

За недомовленістю використовуються тонкі лінії (`\thinlines`). Перемикання на товсті лінії здійснюється командою `\thicklines`. Можна також особисто замовити доречну товщину лінії. Товщина вказується в одиницях довжини TEX'а. Вказати товщину, це те ж саме, що сказати LATEX'у якого розміру пікселі (квадратики, що утворюють рисунок) він повинен використовувати при створенні зображення.

18.2.2 Пакет `curves`

Пакет `curves` расширює набір графічних команд та їх можливостей оточення `picture`. Обговоримо все по порядку.

Можна одержати бажане коло командою

`\bigcircle[`*аргумент заповненості*`]{`*діаметр*`}`

Необов'язковий параметр регулює заповненість точками. До речі, можна розташувати коло безпосередньо в тексті. Це коло одержане командою `\bigcircle[6]{24}`. Як бачите його центр лежить на базовій лінії.

У більшості команд пакета `curves` призначення параметра `[`*аргумент заповненості*`]` такий же, як і у прикладі.

Дуги кола кресляться командою

`\arc[`*аргумент заповненності*`](`$x_{початкове}$`,`$y_{початкове}$`){`*кут в градусах*`}`

```
\frame{ \begin{picture}(60,35)
\put(30,30){\arc(-30,0){120}}
\end{picture} }
```

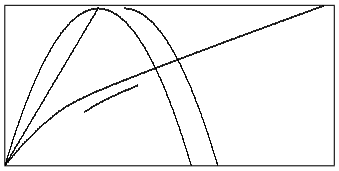
➢Вправа 18.2

В яких координатах задається початкова точка дуги? В якому напрямі креслиться дуга(по часовій чі проти)? З'ясуйте експериментально.

Задача про побудову кривої, що проходить через задані точки, розв'язується командою

`\curve[`*аргумент заповненості*`]`$(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$

Кінці кривої розташовані в початковій і кінцевій точках. Нагадаємо, що в загальному випадку для двох точок це пряма, для трьох — парабола.



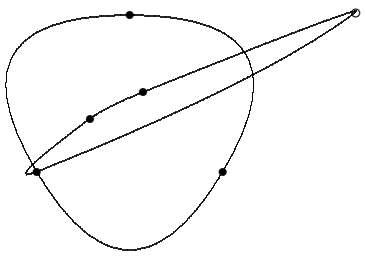
```
\frame{ \begin{picture}(120,60)
\curve(0,0,35,59)
\curve(0,0,35,59,70,0)
\curve(0,0,20,20,40,30,120,60)
\put(10,0){\tagcurve(0,0,35,59,70,0)
\tagcurve(0,0,20,20,40,30,120,60)}
\end{picture} }
```

В цьому прикладі накреслені частини дуг кривих. Дуги кресляться від другої до передостанньої точки (при трьох точках — до останньої) командою

`\tagcurve[`*аргумент заполненности*`]`$(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$

Замкнена гладка крива, щопроходить через вказані точки, будується командою

`\closecurve[`*аргумент заповненості*`]`$(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$



```
\begin{picture}(130,80)(-10,-20)
\closecurve(0,0,35,59,70,0)
\closecurve(0,0,20,20,40,30,120,60)
\put(0,0){\circle*{3}}
\put(35,59){\circle*{3}}
\put(70,0){\circle*{3}}
\put(20,20){\circle*{3}}
\put(120,60){\circle{3}}
\put(40,30){\circle*{3}}
\end{picture}
```

Пакет curves надає дуже важливу команду, яка дозволяє претворювати будть-який рисунок за допомогою матриці розміру 2×2. Така матриця задає афінне перетворення площіни, зокрема так можно задавати обертання. Це команда

`\scaleput(`x,y`){`*елементи рисунка*`}`

Команда `\scaleput` діє подібно до команди `\put`, тільки перед тем як розташувати об'єкт, застосовує до нього перетворення — множення координат об'єкта на матрицю $\begin{pmatrix} \texttt{\textbackslash xscale} & \texttt{\textbackslash xscaley} \\ \texttt{\textbackslash yscalex} & \texttt{\textbackslash yscale} \end{pmatrix}$.

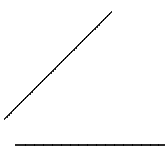
За недомовленістю задається одинична матриця. До тих пір, поки не змінюємо значення матриці, команда `\scaleput` нічим не відрізняється від команди `\put`. Для перевизначення елементів матриці потрібно використати команду `\renewcommand{`*ім'я елемента*`}{`*значення*`}`. Нагадаємо, що дія команди `\renewcomand` обмежена межами групи, що можна використовувати для одержання композицій перетворень.

➢Вправа 18.3

Припустимо, що в одному рисунку змінили значення матриці команди `\scaleput`. Чи потрібно відновлювати одиничну матрицю в іншому рисунку?

```
\begin{picture}(120,85)(0,-20)
\renewcommand{\yscale}{.5}
\curvedashes[1mm]{0,1.5,1.2}
\scaleput(0,0){\put(30,30){ \bigcircle{55}}}
\end{picture}
```

Приклад обертання на 45^o:



```
\begin{picture}(120,85)(-40,-20)
\scaleput(10,10){\curve(0,0,40,40)}
\renewcommand{\xscale}{.7}
\renewcommand{\xscaley}{.7}
\renewcommand{\yscalex}{-.7}
\renewcommand{\yscale}{.7}
\scaleput(10,10){\curve(0,0,40,40)}
```


```
\end{picture}
```

Об'єкти пакета curves можна креслити не суцільною лінією, а пунктирною. Описують вигляд пунктирної лінії командою

`\curvedashes[`*одиниця довжини*`]{`*0, довжина рисочки, довжина пропуску*`}`

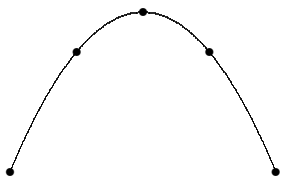
Для відновлення пунктирної лінії, яка задана за недомовленістю, використовують команду `\curvedashes{}`. Инколи виникає необхідність зробити підписи чи поставити мітки вздовж створюваних кривих. Підписи розташовуються командою

`\curvesymbol{`*об'єкт*`}`

Тоді наступна за нею команда креслення кривої буде разташовувати об'єкт команди `\curvesymbol`, а не креслити саму криву. Для команди малювання об'єктів можна вказати кількість символів, які треба розмістити, для кожної дуги кривої. Кількість вказується як від'ємне число у необов'язковому аргументі команди креслення кривої. Подивіться приклад — він прояснить сказане.


```
\begin{picture}(120,85)(0,-20)
\curvesymbol{a}
\put(30,30){\curve[-3](0,0,20,20)}
\put(30,30){\curve(0,0,20,20)}
\end{picture}
```

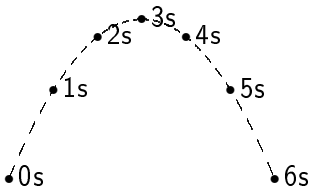
В наступному прикладі нарисуємо параболу і розташуємо на ній по три точки на дугу. Зверніть увагу, як центруються точки за допомогою фантома.



```
\begin{picture}(120,85)(0,-20)
\curvesymbol{\phantom{\circle*{3}}\circle*{3}}
\put(0,0){\curve(0,0,50,60,100,0)}
\put(0,0){\curve[-2](0,0,50,60,100,0)}
\end{picture}
```

➢Вправа 18.4

Отримайте рисунок, що зображений справа.
Вказівка. Скористайтеся прикладом, що наведений вище. Використайте лічильник для створення підпису.



18.3 В рамках

Виділення в рамки підвищує читабельність документа. В найпростішому випадку можна брати блок в рамку за допомогою команди `\fbox{`*блок*`}`. Пакет fancybox розширює можливості створення рамок. Почнемо із описання типів рамок, що доступні в цьому пакеті.

18.3.1 Декоративні рамки

Звичайна рамка `\fbox`: Для цієї і решти рамок відстань між текстом і рамкою задається параметром `\fboxsep` (за недомовленістю цей параметр дорівнює 3 pt). Товщина рамки задається параметром `\fboxrule` (0.4 pt).

Звичайна рамка `\frame`: Для цієї рамки параметр `\fboxsep` доівнює 0.

Рамка з тінню `\shadowbox`: Товщина рамки задається параметром `\fboxrule` (0.4 pt). Ширина тіні `\shadowsize` (4 pt).

Звичайна рамка `\doublebox`: Товщина внутрішньої і зовнішньої рамок дорівнюють відповідно .75`\fboxrule` і 1.5`\fboxrule`. Відстань між рамками — 1.5`\fboxrule` plus .5 pt.

В рамках `\ovalbox`: Діаметр скруглюючих дуг задаємо за допомогою команди `\cornersize{`*число*`}` (0.5). За цією командою діаметр встановлюється рівним числу, помноженому на мінімум довжини чи висоти рамки. Діаметр можна вказати явно командою `\cornersize*{`*довжина*`}`.

В рамках `\Ovalbox`: Така ж, як `\ovalbox`, але товстішою рамкою.

18.3.2 Зберегти блок

Можна сформувати блок, а потім багаторазово використовуати його в тексті. Блок можна розуміти як змінну. Спочатку потрібно об'явити ім'я блока

`\newsavebox{`*ім'я-команда*`}`

Очевидно, що це ім'я команди не повинно бути визначене раніше. Визначена так команда має глобальну область дії. Надається значення ж за допомогою довгої команди або її скорочення

`\savebox{`*ім'я-команда*`}[`*ширина*`][`*позиціонування текста в блоці l/r/c/s*`]{`*текст*`}`
`\sbox{`*ім'я-команда*`}{`*текст*`}`

Друк заповненого блока здіснюється командою

`\usebox{`*ім'я блока*`}`

-a-a -a-A

```
\newsavebox{\biga}
\sbox{\biga}{\Huge -a}
\usebox{\biga}\usebox{\biga}
\usebox{\biga}-A
```

Відмітимо, що в деяких випадках, наприклад для дослівного відтворення, зручніше використовувати аналог команди `\sbox` — оточення `lrbox`:

`\beginlrbox{`*ім'я блока*`}` текст `\endlrbox`

18.3.3 Рамки-оточення fancybox

В пакеть fancybox існує аналог згаданої вище команди для збереження блока — оточення `Sbox`. Відміна полягає в тому, що ім'я блока вказувати не потрібно, блок знаходиться командою `\TheSbox`.

Для взяття оточень в рамки визначені їх их рамочн аналоги. Це оточення вирівнювання `Bflushleft`, `Bflushright`, `Bcenter` та оточення переліку `Bitemize`, `Benumerate`, `Bdescription`. Для оточення формул `Beqnarray`,

Beqnarray* створюється не відцентрований блок, рамку для якого можн добавити самому, при цьому рамка буде впритул до країв формули. Ці команди можна використовувати для створення власних оточень.

```
\newenvironment{FramedEqn}%
 {\setlength{\fboxsep}{15pt}
   \setlength{\mylength}{\linewidth}%
   \addtolength{\mylength}{-2\fboxsep}%
   \addtolength{\mylength}{-2\fboxrule}%
   \Sbox
   \minipage{\mylength}%
      \setlength{\abovedisplayskip}{0pt}%
      \setlength{\belowdisplayskip}{0pt}%
      \equation}%
 {\endequation\endminipage\endSbox
   \[\fbox{\TheSbox}\]}
 \begin{FramedEqn}
   x & = & y\\
   y & > & x \\
   \int_4^5 f(x)dx & = & \sum_{i\in F} x_i
 \end{FramedEqn}
```

$$x = y \tag{5}$$
$$y > x \tag{6}$$
$$\int_4^5 f(x)dx = \sum_{i\in F} x_i \tag{7}$$

У пакета fancybox широкі можливості обертання блоків. Наприклад, визначена команда \LandScape, яка дозволяє змінити орієнтацію сторінок. Більш детально про це можна прочитати в документації до пакету.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in roman refer to the pages where the entry is used.