Fakultas Sains dan Teknologi
Universitas Islam Negeri Walisongo Semarang

PENERBIT
LITTLE SOLEIL

Buku ini diperuntukkan sebagai panduan bagi para pemula, berisikan ilmu-ilmu dasar LaTeX, yakni bekal-bekal yang sekiranya diperlukan dalam pembuatan dokumen-dokumen dan media presentasi sederhana.

ISBN 978-623-97183-9-8

Alamat Kantor Penerbit Little Soleil :

Perum Pesona Bumi Mandiri 2 RT 6, RW 3 Kav. 61
Jl. Anggrek, Beran, Tambaharjo, Pati, Jawa Tengah
59119

Irman Said Prastyo | PANDUAN LATEX UNTUK PEMULA | Penerbit Little Soleil

Panduan LaTeX untuk Pemula

Irman Said Prastyo

HIBAH FAKULTAS SAINS DAN TEKNOLOGI UIN WALISONGO SEMARANG 2021

PANDUAN

LaTeX

UNTUK PEMULA

Irman Said Prastyo

PANDUAN LaTeX UNTUK PEMULA

Penulis : Irman Said Prastyo

Perancang sampul : Yoyok Dwi Prastyo
Tata letak : Yoyok Dwi Prastyo

Penerbit Little Soleil
Perum Pesona Bumi Mandiri 2 RT 6, RW 3 Kav. 61
Jl. Anggrek, Beran, Tambaharjo, Pati, Jawa Tengah
59119

ISBN : 978-623-97183-9-8

Karya sederhana ini penulis persembahkan
untuk ibu dan ayah

Kata Pengantar

Puji syukur yang setinggi-tingginya hanyalah untuk Dia yang punya alam dan yang menciptakan kehidupan. Atas izinNya pula penulisan buku ini dapat terselesaikan sebab tidak ada satupun perkara yang luput dari keterlibatanNya.

Ucapan terima kasih terhatur untuk segenap pihak yang turut memberikan dukungan kepada penulis selama proses penyusunan buku ini:

- Ibu, ayah, cahaya bagi penulis yang terus mendoakan demi kesehatan dan kelancaran segala urusan penulis.
- Bapak Dr. H. Ismail, M.Ag. (dekan FST UIN Walisongo), Bapak Dr. Saminanto, M.Sc. (wakil dekan 1), Bapak Dr. H. Nur Khoiri, M.Ag., (wakil dekan 2), dan Ibu Dr. Hj. Nur Khasanah, M.Kes. (wakil dekan 3) yang telah memberikan dukungan terutama dalam bentuk kebijakan hingga pembiayaan untuk penerbitan buku ini.
- Bapak Joko Budi Poernomo, M.Pd. (kajur fisika), Bapak Edi Daenuri Anwar, M.Si. (sekjur fisika), Bapak Agus Sudarmanto, M.Si. (kaprodi fisika), dan Bapak Muhammad Izzatul Faqih, M.Pd. (sekprodi fisika), yang telah membimbing dan mengarahkan penulis.
- Teman-teman dosen KBK fisika teori, Pak Ardhi, Bu Arsini, Mbak Isti, dan Mbak Wirdah, yang telah *mensupport* dan terutama sekali telah bersama-sama mengadakan Pelatihan LaTeX 2021 sehingga menjadi pemicu awal bagi penulisan buku ini.

- Teman-teman dosmud fisika UIN Walisongo, Mbak Susi, Mbak Heni, Mbak Affa, Mbak Qisthi, Mbak Sheilla, Mas Hartono, dan Mas Rian. Semua seperti keluarga.

- Teman-teman dosen fisika yang tidak bisa penulis sebutkan satu per satu.

- Teman-teman Humas FST UIN Walisongo, Mas Hadi, Mas Ghani, Mas Minan, Mas Mughis, Kak Lenni, Mbak Rina, Mbak Galih, dan Teh Nissa. Kalian "rumah" tersendiri bagi penulis.

- Dr.rer.nat. Muhammad Farchani Rosyid dan Dr. Dwi Satya Palupi, "orang tua" sekaligus penyemangat semenjak masih di bangku kuliah hingga setelah lulus dari UGM.

- Teman-teman alumni UGM, Yusuf, Zam, Iqbal, Qosim, Rahmat, Mas Eko, Risal, Puri, Ita, Mbak Wulan, yang dari mereka penulis banyak belajar ilmu-ilmu baru, pelajaran-pelajaran hidup, dan juga ilmu tentang LaTeX yang menjadi "modal" bagi penulisan buku ini.

- Pak Yoyok Dwi Prastyo, guru sekaligus teman yang darinya banyak nasehat untuk kebaikan. Semoga sehat selalu.

Muara dari setiap perjalanan, pencarian ilmu dan segala proses belajar adalah kebermanfaatan. Begitu penulis mempercayainya. Ilmu apapun adalah pintu untuk pengenalan dan jalan untuk "melihat". Semoga Dia mengantarkan kita untuk lebih tajam "melihat" dan semoga buku ini menjadi sarana untuk saling menebar manfaat.

Buku ini dimaksudkan sebagai panduan bagi para pemula, berisikan ilmu-ilmu dasar LaTeX, yakni bekal-bekal yang sekiranya diperlukan bagi pembuatan dokumen-dokumen dan media presentasi sederhana. Tentu sangat disadari oleh penulis bahwa ilmu tentang LaTeX teramat luas, jauh dari yang termuat di dalam buku ini.

Berbagai pihak telah berinisiatif mengembangkan LaTeX sehingga LaTeX menjadi semakin *powerfull*, terbukti dengan hadirnya banyak *package* baru yang memperkaya perbendaharaan fungsi dan "kemampuan" LaTeX itu sendiri. Bahkan, jika seluruh *package*

hendak dibicarakan, masing-masingnya tidaklah kurang untuk dibuat lagi menjadi buku-buku baru. Oleh karenanya, buku ini hanya membahas beberapa hal yang oleh penulis dianggap penting.

Dari tidak sempurnanya karya, penulis memohon maaf kepada seluruh pihak, utamanya pembaca, jika terdapat kekurangan-kekurangan atau bahkan kesalahan-kesalahan di dalam penyampaian isi buku ini. Koreksi dan masukan yang bersifat membangun sangat penulis butuhkan untuk perbaikan ke depan. Akhir kata semoga kita semua dimudahkan urusan dan selalu dalam keadaan berbahagia.

Pati, 3 September 2021

Penulis

Daftar Isi

Kata Pengantar **5**

Daftar Isi **9**

1 Sekilas tentang LaTeX **13**
1.1 Gambaran Umum tentang Perintah 13
1.2 Cara-cara Alternatif 14

2 Dasar-dasar Pembuatan Dokumen **19**
2.1 Struktur Dasar Dokumen 19
2.2 Membuat Judul 21
2.3 Membuat Abstrak 22
2.4 Membuat Hirarki: Bab, Subbab, dst. 22
2.5 Pindah Halaman 23
2.6 Perataan Baris 24
2.7 Pemenggalan Kata 25
2.8 Pindah Baris dan Paragraf serta Spasi 26
2.9 Teks Tebal, Miring dan Bergaris Bawah 28
2.10 Ukuran Font 29
2.11 Jenis Font 29
2.12 Membuat Daftar dan Penomoran 30
2.13 Catatan Kaki 33
2.14 Teks Multi Kolom 33
2.15 Mewarnai Teks 34
2.16 Mengatur Penomoran Halaman 35
2.17 Membuat Daftar Isi 36

2.18 Membuat Hyperlink 37

3 Pengetikan Teks Matematis 39
3.1 Mode *Inline* dan *Displaymath* 39
3.2 Spasi 41
3.3 Titik-titik 41
3.4 *Subscript* dan *Superscript* 41
3.5 Pecahan 42
3.6 Akar 42
3.7 Karakter-karakter Khusus 43
3.8 Huruf-huruf Yunani 45
3.9 Karakter-karakter Khusus dengan Batas 46
3.10 Anak Panah 47
3.11 Fungsi-fungsi Matematis 48
3.12 Vektor 49
3.13 Aksen-aksen Matematis 49
3.14 Matriks 51
3.15 Kurung Berketinggian Sesuai Karakter 52
3.16 Teks Matematis Beberapa Baris 53
3.17 Pelabelan Persamaan 54
3.18 Teks Matematis Tebal dan Miring 56
3.19 Ubah Font Matematis 56
3.20 Teks Biasa di Dalam Teks Matematis 57

4 Menambahkan File Gambar 59
4.1 Cara Sederhana Memasukkan Gambar 59
4.2 *Environment* `figure` dan Letak Gambar 63
4.3 Skala, Ukuran Gambar, dan Rotasi 67
4.4 Menambahkan *Caption* 70
4.5 Menambahkan Label Gambar 72
4.6 Memasukkan Gambar dari Suatu Folder 73

5 Menambahkan Tabel 77
5.1 Membuat Tabel Sederhana 77
5.2 Letak Tabel dan *Caption* 80
5.3 *Text Wrapping* pada Tabel 81
5.4 Penggabungan Baris dan Kolom 83
5.5 Perataan Teks Satu Sel 88

5.6 Mengubah Tinggi Baris 89
5.7 Tabel Profesional 92
5.8 Modifikasi Garis pada Tabel 93
5.9 *Environment* `array` 95

6 Menggambar dengan Ti*k*Z **97**
6.1 Kanvas dan Sistem Koordinat 98
6.2 Penulisan Koordinat 99
6.3 Penamaan Titik dan Pelabelan 100
6.4 Mengatur Letak Gambar 101
6.5 Menggambar Garis dan Poligon 102
6.6 Modifikasi Gambar 105
6.7 Menggambar Persegi Panjang 109
6.8 Menggambar Lingkaran 110
6.9 Menggambar Elips 111
6.10 Menggambar Busur Lingkaran 112
6.11 Menggambar Busur Elips 114
6.12 Menambahkan Teks pada Gambar 115
6.13 Sekilas Mengenal PGFPlots 118

7 Membuat Presentasi dengan Beamer **123**
7.1 Sekilas tentang Beamer dan Efek 123
7.2 Keuntungan Menggunakan Beamer 124
7.3 Memulai Pembuatan Presentasi 125
7.4 *Overlay Pause* 132
7.5 *Overlay Specifications* pada Daftar 135
7.6 Beberapa *Overlay specifications* Lainnya 141

Bibliografi **149**

Bab 1

Sekilas tentang LaTeX

1.1 Gambaran Umum tentang Perintah

Secara umum (tapi tidak semua), perintah pada LaTeX diawali dengan *backslash* (\). Beberapa perintah ada yang memerlukan *argument*. Ada dua macam *argument*, yakni *mandatory* (diapit oleh kurung kurawal {...}) dan *optional* (diapit oleh kurung kotak [...]).

Contoh:

1. `\documentclass[a4paper]{article}`
2. `\textit{Ini tulisan miring}`
3. `\bfseries`

Secara khusus, setiap perintah memiliki aturan penulisannya sendiri-sendiri, tentang apa yang wajib disertakan dan lain-lain, yang tidak akan disampaikan satu per satu di sini. Beberapa perintah juga ada yang aturan penulisannya memang tidak menggunakan *backslash* (\).

Untuk mengkhususkan perintah hanya pada suatu bagian teks, bagian itu dapat dikelompokkan dengan cara diapit menggunakan kurung kurawal {...}.

Contoh:
`{\bfseries Ini tulisan tebal} dan yang ini tidak tebal.`

Perintah `\bfseries` (untuk menebalkan teks) di atas hanya berlaku untuk tulisan yang ada di dalam kurung kurawal. Hasilnya sebagai berikut:

Ini tulisan tebal dan yang ini tidak tebal.

Beberapa karakter khusus tidak dapat ditulis secara langsung menggunakan karakter-karakter itu sebab karakter-karakter itu telah berfungsi menjadi bagian dari perintah. Misal, kita tidak dapat menulis karakter-karakter berikut secara langsung: \, {, &, $, _, ^, dan %. Karakter-karakter ini berturut-turut harus ditulis dengan perintah berikut: `\textbackslash,` `\{,` `\&,` `\$,` `\_,` `\^{},` dan `\%`.

Penting juga untuk diketahui bahwa penggunaan `%` berfungsi untuk membuat teks tidak terbaca saat *dicompile*. Teks yang ditulis setelah `%` tidak ikut dimunculkan sebagai teks pada dokumen yang dihasilkan. Perintah ini berfungsi sekedar untuk memberikan catatan-catatan atau keterangan.

Contoh:
`\chapter{Fisika Itu Seru} %Ini untuk membuat bab`

Tulisan `Ini untuk membuat bab` tidak akan muncul dalam dokumen.

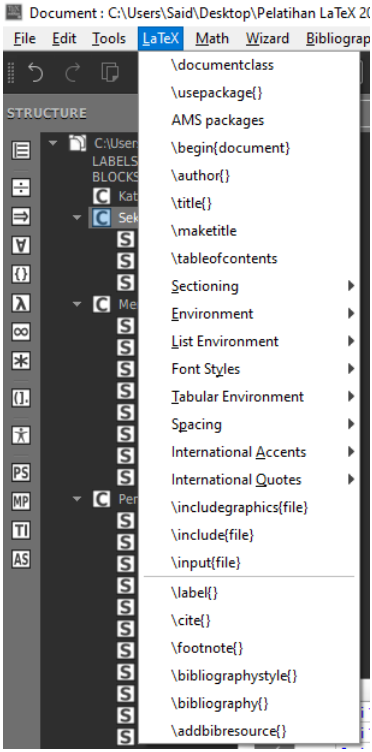
1.2 Cara-cara Alternatif

Pertanyaan mendasar yang mungkin muncul di fikiran kita adalah "Apakah kita harus menghafalkan semua perintah yang ada di LaTeX?" Tentu jika kita mampu menghafalkan seluruhnya akan lebih baik, akan tetapi jika kita tidak mampu menghafalkan seluruhnya, berbagai LaTeX editor telah menyediakan cara yang lebih mudah yaitu dengan hanya melakukan klik tombol.

Tidak mungkin seluruh menu yang ada di berbagai LaTeX editor dijelaskan rinci di sini. Hanya sebagai gambaran, berikut akan

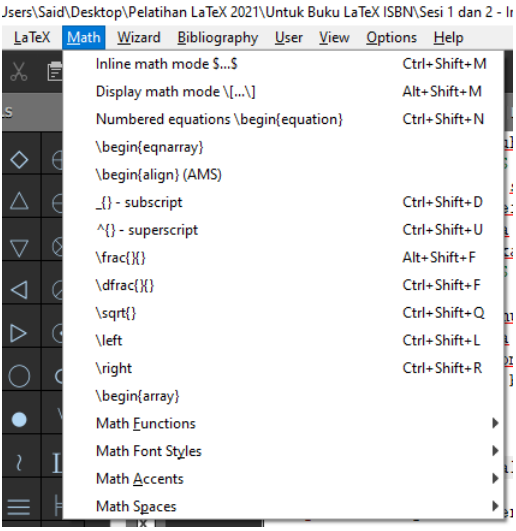
ditunjukkan beberapa cara cepat menulis perintah jika kita menggunakan TEX maker.

Berbagai perintah umum yang (mungkin) sulit kita hafal sudah terangkum dan dikelompok-kelompokkan di menu bar (deretan menu bagian atas). Sekumpulan perintah terkait pembuatan dokumen dan lain-lain terdapat di menu LaTeX (gambar 1.1), perintah-perintah matematis terdapat di menu Math (gambar 1.2), dan seterusnya. Silakan dijelajahi sendiri untuk melihat lebih banyak.



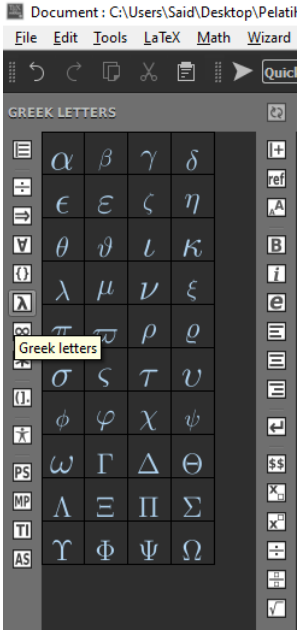
Gambar 1.1: Tampilan menu LaTeX pada menu bar

Kita juga dapat memanfaatkan menu-menu yang ada di sisi kiri. Agar mahir menggunakan, silakan dijelajahi (dicoba-coba) sendiri!



Gambar 1.2: Tampilan menu Math pada menu bar

Sebagai contoh, kumpulan perintah untuk menuliskan huruf-huruf Yunani terdapat di menu dengan icon huruf lambda (λ) (gambar 1.3).



Gambar 1.3: Tampilan menu untuk menuliskan huruf Yunani

Bab 2

Dasar-dasar Pembuatan Dokumen

Dasar-dasar pembuatan dokumen yang dijelaskan di bab ini adalah yang sifatnya umum (sering digunakan) dan tidak benar-benar detail. Prinsipnya, bab ini disajikan (sekedar) agar pembaca mengerti gambaran besar pembuatan dokumen menggunakan LaTeX dan kemudian dapat digunakan untuk membantu pembaca dalam menyelesaikan pekerjaan-pekerjaannya.

2.1 Struktur Dasar Dokumen

Pembuatan dokumen LaTeX **wajib** dimulai dengan penulisan kelas dokumen di awal, yakni

`\documentclass[...]{kelas dokumen}`

misal `\documentclass[a4paper,12pt]{article}`. *Optional argument* tidak wajib disertakan tetapi *mandatory argument*, yakni `kelas dokumen`, **wajib** dituliskan. Kelas dokumen di antaranya adalah `article, book, report, letter` dan `beamer`. *Optional argument* bisa berisi beberapa opsi yang penulisannya dipisahkan dengan ko-

ma (,). Beberapa opsi itu di antaranya ukuran kertas (`a4paper, a5paper, b5paper, letterpaper,` atau `legalpaper`), ukuran huruf (`10pt, 11pt,` atau `12pt`), layout (`oneside` atau `twoside`), teks dua kolom (`twocolumn`), dan halaman judul (`titlepage`). Halaman judul di sini maksudnya adalah halaman judul di awal yang terpisah dan penomoran halaman dimulai dari halaman setelahnya.

Bagian berikutnya adalah *preamble*. Di sini dapat dituliskan berbagai *package* yang membuat perintah-perintah tertentu dapat bekerja. Dalam hal ini beberapa perintah adalah bawaan *package* tertentu. Bukanlah sebuah kewajiban untuk menuliskan suatu *package* di *preamble* jika perintah-perintah bawaannya tidak hendak digunakan. Beberapa *package* yang sering digunakan adalah `geometry` (untuk mengatur margin), `babel` (untuk pemilihan bahasa), `amsmath` (untuk penulisan matematis), dan `graphicx` (untuk menginput gambar). Cara memanggil *package* adalah dengan menggunakan `\usepackage{nama package}`. Beberapa di antaranya juga menggunakan *optional argument*.

Contoh:

```
\usepackage[left=2cm, right=2cm, bottom=3cm, top=3cm]
{geometry}

\usepackage[bahasa]{babel}
\usepackage{amsmath}
\usepackage{graphicx}
```

Catatan: meskipun pemanggilan *package* di atas hanya contoh, tetapi penting untuk sekaligus diperhatikan pemanggilan *package* pertama dan kedua, yakni `geometri` dan `babel`. Tidak ada bahasan lebih lanjut tentang dua *package* ini ke belakang, tetapi dalam pembuatan dokumen keduanya sangat sering digunakan. Penting juga untuk diperhatikan bahwa kode `[bahasa]` yang dituliskan sebagai *optional argument* dari *package* `babel` merujuk ke bahasa Indonesia. Setiap bahasa ada kodenya tersendiri.

Bagian berikutnya adalah isi dokumen. Isi dokumen **selalu** diawali dengan `\begin{document}` dan diakhiri (ditutup) dengan `\end{document}`.

Contoh:

```
\begin{document}
Di sini adalah tempat menuliskan isi dokumen
\end{document}
```

Di dalam isi dokumen bisa dituliskan beberapa bagian dengan suatu perintah khusus yang dimulai dengan `\begin{...}` dan diakhiri dengan `\end{...}` (titik-titik diisi sesuai perintah yang diinginkan). Bagian semacam ini disebut sebagai *environment*.

Sebagai penutup subbab ini, agar aturan peletakan struktur dokumen yang telah kita bahas menjadi lebih jelas, perhatikan contoh keseluruhan kode yang bisa kita tuliskan untuk membuat sebuah dokumen sederhana (misal dokumen jenis artikel yang ditulis pada kertas A4) berikut ini!

Contoh:

```
% Kelas dokumen:
\documentclass[a4paper]{article}

% Preamble:
\usepackage[left=2cm, right=2cm, bottom=2cm, top=2cm]
{geometry}
\usepackage[bahasa]{babel}

% Isi dokumen:
\begin{document}
Ini teks dokumen sederhana
\end{document}
```

2.2 Membuat Judul

Kita bisa saja menuliskan judul di dokumen secara manual dengan mengatur teks agar menjadi rata tengah, ukuran font lebih besar, dan lain sebagainya (cara-caranya akan dibahas kemudian). Akan tetapi, LaTeX sendiri telah memfasilitasi perintah untuk pembuatan judul. Penulisan judul diawali dengan pembuatan nama judul,

nama penulis, dan tanggal menggunakan perintah berikut:

```
\title{Nama Judul}
\author{Nama Penulis}
\date{Tanggal}
```

Perintah di atas dapat dituliskan baik itu sebelum atau setelah `\begin{document}`. Judul (beserta nama penulis dan tanggal) baru akan muncul setelah dipanggil di dalam dokumen menggunakan perintah `\maketitle`. Jika di awal saat menuliskan kelas dokumen (perintah paling atas) kita menyertakan `titlepage` di *optional argument* maka nantinya judul yang muncul adalah pada halaman awal yang terpisah dari halaman-halaman lain. Penulis tidak memberikan contoh di subbab ini. Untuk melihat hasil, silakan dicoba sendiri!

2.3 Membuat Abstrak

Perintah untuk membuat abstrak hanya dimengerti untuk kelas dokumen `article` dan `letter`. Cara membuat abstrak yaitu dengan memasukkan teks (isi abstrak) di dalam *environment* `abstract` (lengkapnya yaitu `\begin{abstract}...\end{abstract}`) yang dibuat setelah `\begin{document}`.

Abstrak dimunculkan di dokumen dengan judul "Abstract" (jika bahasanya adalah bahasa Inggris) dan "Ringkasan" (jika bahasa telah diubah ke bahasa Indonesia menggunakan *package* `babel`). Jika menghendaki judulnya sesuai dengan keinginan kita, misal kita ingin membuat judulnya adalah "Abstrak" (atau yang lain), sebelum menuliskan *environment* `abstract` tuliskan terlebih dahulu `\renewcommand{\abstractname}{Abstrak}`.

2.4 Membuat Hirarki: Bab, Subbab, dst.

Untuk membuat bab, subbab, sub-subbab, sub-sub-subbab, dan seterusnya kita tidak perlu menuliskan penomoran dan letaknya secara manual. Cukup kita gunakan perintah-perintah yang disedia-

kan oleh LaTeX dan penomoran akan diatur dengan sendirinya oleh LaTeX. Perintah dalam hirarki ini ada tujuh level yang disediakan, yaitu meliputi (setiap level adalah sublevel dari level sebelumnya):

1. `\part{...}`
   (tidak tersedia untuk kelas dokumen `letter`)
2. `\chapter{...}`
   (hanya tersedia untuk kelas dokumen `book` dan `report`)
3. `\section{...}`
   (tidak tersedia untuk kelas dokumen `letter`)
4. `\subsection{...}`
   (tidak tersedia untuk kelas dokumen `letter`)
5. `\subsubsection{...}`
   (tidak tersedia untuk kelas dokumen `letter`)
6. `\paragraph{...}`
   (tidak tersedia untuk kelas dokumen `letter`)
7. `\subparagraph{...}`
   (tidak tersedia untuk kelas dokumen `letter`)

Titik-titik diisi dengan teks sesuai judul, subjudul, dan sebagainya. Perintah-perintah ini dituliskan di antara `\begin{document}` dan `\end{document}`. Posisi dan perataan teks yang mengikuti (ditulis setelah) perintah-perintah ini juga diatur dengan sendirinya oleh LaTeX.

Tidak ada contoh untuk subbab ini, tetapi pembaca dapat memperhatikan penulisan judul bab dan subbab yang ada di buku ini. Keseluruhannya ditulis menggunakan dua dari perintah-perintah di atas, yaitu `\chapter` dan `\section`.

2.5 Pindah Halaman

Perintah untuk berpindah halaman adalah `\newpage`. Teks yang ditulis setelah perintah ini akan diletakkan di halaman baru (halaman setelahnya) meskipun teks di halaman awalnya belum memenuhi halaman.

2.6 Perataan Baris

Dokumen LaTeX secara *default* tanpa pengaturan khusus berada pada mode *justify* (rata kiri kanan). Pengaturan dapat diubah ke rata kiri, rata kanan atau rata tengah dengan tiga *environment* berikut:

1. Rata kiri:
   ```
   \begin{flushleft}
   Teks yang diatur rata kiri
   \end{flushleft}
   ```
2. Rata kanan:
   ```
   \begin{flushright}
   Teks yang diatur rata kanan
   \end{flushright}
   ```
3. Rata tengah:
   ```
   \begin{center}
   Teks yang diatur rata tengah
   \end{center}
   ```

Cara lain juga dapat digunakan untuk perataan baris, yaitu dengan perintah `\raggedright` (rata kiri), `\raggedleft` (rata kanan) dan `\centering` (rata tengah). Perintah ini akan berlaku untuk semua teks yang ditulis setelahnya.

Contoh:
Perhatikan teks berikut!

Seismograf adalah suatu alat pencatat (pendeteksi) kekuatan gempa.

Seismograf dibuat dalam berbagai macam bentuk dengan komponen utamanya adalah bandul dan/atau pegas.

Sebuah seismograf secara khusus didesain hanya untuk mencatat getaran dalam satu arah.

Pencatatan getaran ke berbagai arah secara bersamaan memerlukan beberapa buah seismograf yang digunakan serentak.

Teks di atas ditulis menggunakan kode berikut (di antara \begin {document} dan \end {document}):

```
Seismograf adalah suatu alat pencatat (pendeteksi)
kekuatan gempa.
\begin{flushleft}
Seismograf dibuat dalam berbagai macam bentuk dengan
komponen utamanya adalah bandul dan/atau pegas.
\end{flushleft}
\begin{flushright}
Sebuah seismograf secara khusus didesain hanya untuk
mencatat getaran dalam satu arah.
\end{flushright}
\begin{center}
Pencatatan getaran ke berbagai arah secara bersamaan
memerlukan beberapa buah seismograf yang digunakan
serentak.
\end{center}
```

2.7 Pemenggalan Kata

Yang dilakukan LaTeX terhadap teks pada mode *justify* agar teks menjadi rata kiri dan kanan adalah dengan memenggal kata (menggunakan strip) jika panjangnya kata itu melewati lebar teks yang semestinya. Kita mungkin berfikir bahwa dengan mengatur bahasa menjadi bahasa Indonesia (dengan *package* `babel`) telah cukup untuk membuat LaTeX mengerti aturan pemenggalan kata bahasa Indonesia yang baik dan benar. Kabar "buruknya", sangkaan ini tidak benar. LaTeX belum "berdamai" dengan setiap bahasa. Ini kekurangan yang harus jujur disebutkan.

Untuk mengatasi pemenggalan kata yang salah, misalkan kata "langkah" dipenggal menjadi "lan-gkah" dan bukan "lang-kah", kita perlu menambahkan *package* `hyphenat`. Tuliskan di *preamble* \usepackage{hyphenat}! Selanjutnya kita buat sendiri daftar pemenggalan kata yang benar untuk kata-kata yang kita temukan salah pemenggalannya di akhir baris teks. Di dalam daftar, antar kata dipisahkan dengan spasi. Cara membuat daftar adalah pada *pre-*

amble (setelah *package* `hyphenat`) tuliskan `\hyphenation{daftar pemenggalan kata}`.

Contoh:

```
\hyphenation{pen-da-hu-lu-an lang-kah mi-sal-kan}
```

2.8 Pindah Baris dan Paragraf serta Spasi

Perintah untuk berpindah baris adalah `\\` (dapat disertai *optional argumen* untuk membuat spasi vertikal, misal `\\[2cm]`) dan untuk berpindah atau membuat paragraf baru adalah dengan menekan enter dua kali. Ciri dari paragraf baru adalah tulisan yang dihasilkan jadi menjorok ke dalam di awal paragraf itu, **kecuali paragraf pertama** yang secara *default* oleh LaTeX memang dibuat tidak menjorok. Perpindahan baris dan paragraf juga dapat disertai dengan perintah untuk membuat spasi vertikal menggunakan `\vspace{lebar spasi}` (contoh: `\vspace{2cm}`). Spasi horisontal dibuat dengan perintah `\hspace{lebar spasi}`.

Contoh:
Perhatikan teks di bawah ini!

Turbin pada kasus tertentu dapat dipandang sebagai sistem mekanis stokastik. Kecepatan sudut turbin dari waktu ke waktu tidak tetap, dipengaruhi oleh torka sistem yang berubah-ubah baik besarnya maupun arahnya. Torka sistem dalam hal ini adalah sumbangan dari torka akibat air atau angin penggerak turbin, gaya gesek antara turbin dengan poros turbin serta gaya yang ditimbulkan oleh gesekan antara turbin dengan (misal) tali atau gir penghubung turbin ke generator.

Jenis turbin yang menjadi perhatian di sini adalah turbin dengan suatu simetri (bentuk) sedemikian rupa sehingga sumbangan torka oleh gaya gravitasi bumi sama dengan nol.

Gravitasi dalam hal ini praktis tidak dapat mengubah kecepatan

sudut sistem.

Berbagai jenis turbin banyak didesain dengan memenuhi simetri ini.

Apabila ketinggian poros menjadi acuan bagi energi potensial akibat gravitasi, yakni disepakati bahwa bidang mendatar melalui poros adalah tempat setiap titik massa memiliki energi potensial sama dengan nol, energi potensial (keseluruhan) dari sistem berupa turbin yang kita bicarakan juga bernilai nol.
Ini diakibatkan oleh energi potensial dari titik-titik massa (bagian-bagian turbin) di atas poros dengan yang di bawah poros praktis saling melenyapkan.

Teks di atas ditulis menggunakan kode berikut:

```
Turbin pada kasus tertentu dapat dipandang sebagai
sistem mekanis stokastik.  Kecepatan sudut turbin dari
waktu ke waktu tidak tetap, dipengaruhi oleh torka
sistem yang berubah-ubah baik besarnya maupun arahnya.
Torka sistem dalam hal ini adalah sumbangan dari torka
akibat air atau angin penggerak turbin, gaya gesek
antara turbin dengan poros turbin serta gaya yang
ditimbulkan oleh gesekan antara turbin dengan (misal)
tali atau gir penghubung turbin ke generator.

Jenis turbin yang menjadi perhatian di sini adalah
turbin dengan suatu simetri (bentuk) sedemikian rupa
sehingga sumbangan torka oleh gaya gravitasi bumi sama
dengan nol.
\\[0.2cm]
Gravitasi dalam hal ini praktis tidak dapat mengubah
kecepatan sudut sistem.
\\[0.2cm]
Berbagai jenis turbin banyak didesain dengan memenuhi
simetri ini.
\vspace{0.5cm}

Apabila ketinggian poros menjadi acuan bagi energi
```

```
potensial akibat gravitasi, yakni disepakati bahwa
bidang mendatar melalui poros adalah tempat setiap
titik massa memiliki energi potensial sama dengan nol,
energi potensial (keseluruhan) dari sistem berupa
turbin yang kita bicarakan juga bernilai nol.
\\Ini diakibatkan oleh energi potensial dari titik-
titik massa (bagian-bagian turbin) di atas poros dengan
yang di bawah poros praktis saling melenyapkan.
```

2.9 Teks Tebal, Miring dan Bergaris Bawah

Teks bercetak tebal dapat dibuat dengan perintah \textbf{Teks yang ingin ditulis tebal}. Cara lain adalah dengan menuliskan \bfseries untuk menjadikan teks setelahnya tebal. Teks miring dapat dibuat dengan perintah \textit{Teks yang ditulis miring}. Teks bergaris bawah dapat dibuat dengan menggunakan perintah \underline{Teks yang ditulis bergaris bawah}.

Contoh:
Teks berikut ini,

Ini normal, **ini tebal**, *ini miring*, dan ini bergaris bawah.

ditulis menggunakan kode

```
Ini normal, \textbf{ini tebal}, \textit{ini miring}, dan
\underline{ini bergaris bawah}.
```

Kita juga dapat membuat tulisan bercetak miring dengan menggunakan perintah \emph{Teks yang ingin ditulis miring}. Akan tetapi ada perbedaan hasil dengan perintah \textit. Perintah \textit akan mengganti jenis font tegak dengan jenis font miring (beda jenis font, meskipun nama font sama), sementara perintah \emph akan memiringkan atau menarik teks yang awalnya tegak sehingga menjadi miring.

2.10 Ukuran Font

Perintah untuk mengubah ukuran font dari ukuran paling kecil hingga ukuran paling besar besar berturut-turut adalah `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\normalsize`, `\large`, `\Large`, `\LARGE`, `\huge`, dan `\Huge`. Teks yang ditulis setelah perintah-perintah ini akan berubah ukuran sesuai perintah. Teks yang diubah ukuran fontnya juga dapat dikelompokkan dengan kurung kurawal.

Contoh:
Tulisan berikut ini,

{\small aku }{\large menjadi }{\LARGE semakin } {\Huge besar}

dibuat dengan kode berikut:

```
{\small aku }{\large menjadi }{\LARGE semakin }
{\Huge besar}
```

2.11 Jenis Font

Ada tiga jenis font standar LaTeX (jika tanpa *package* khusus), yaitu:

1. Roman, ditulis dengan didahului perintah `\rmfamily`
2. Sans serif, ditulis dengan didahului perintah `\sffamily`
3. `Typewriter`, ditulis dengan didahului perintah `\ttfamily`

Teks yang diubah ukuran fontnya juga dapat dikelompokkan dengan kurung kurawal.

Contoh:
Tulisan berikut ini,

`Tulisan ini seperti dihasilkan oleh mesin ketik,` berbeda dengan tulisan yang ini.

dibuat menggunakan kode berikut:

```
{\ttfamily Tulisan ini seperti dihasilkan oleh mesin
ketik, } {\sffamily berbeda dengan tulisan yang ini.}
```

2.12 Membuat Daftar dan Penomoran

Untuk membuat daftar dengan penomoran dapat dilakukan menggunakan *environment* `enumerate`. Di dalam *environment* ini, untuk setiap penomoran (*item*) baru dimulai dengan perintah `\item`.

Contoh:
Daftar dengan penomoran berikut ini,

1. merkurius
2. venus
3. bumi

ditulis dengan *environment*

```
\begin{enumerate}
\item merkurius
\item venus
\item bumi
\end{enumerate}
```

Untuk membuat daftar dengan *bullet*, digunakan *environment* `itemize`. Dengan cara yang sama, setiap *item* baru dimulai dengan perintah `\item`.

Contoh:
Daftar dengan *bullet* berikut ini,

- mars
- jupiter
- saturnus

ditulis dengan *environment*

```
\begin{itemize}
\item mars
\item jupiter
\item saturnus
\end{itemize}
```

Baik di *environmet* `enumerate` maupun `itemize`, kita juga dapat mengganti nomor ataupun *bullet* dengan karakter tertentu yang kita inginkan. Caranya adalah dengan menggunakan *optional argument* setelah penulisan `\item`. *Optional argument* di dalam kurung kotak berisi karakter yang diinginkan.

Contoh:
Daftar dengan karakter @ berikut ini,

@ uranus

@ neptunus

@ pluto

ditulis dengan *environment*

```
\begin{itemize}
\item[@] uranus
\item[@] neptunus
\item[@] pluto
\end{itemize}
```

Kita juga dapat berkreativitas memodifikasi nomor-nomor di daftar sesuai dengan *style* kita sendiri melalui penggunaan *package* `enumerate`. Di bagian preamble tuliskan `\usepackage{enumerate}`! Cara membuat daftar dengan penomoran masih sama, yakni menggunakan *environment* `enumerate`. Hal yang membedakan dari sebelumnya adalah kita dapat menyertakan *optional argument* pada perintah `\begin{enumerate}` yang berisikan *style* penomoran yang ingin kita buat.

Contoh:
Daftar dengan *style* penomoran berikut ini,

(A-1) fisika

(A-2) kimia

(A-3) biologi

ditulis dengan *environment*

```
\begin{enumerate}[({A}-1)]
\item fisika
\item kimia
\item biologi
\end{enumerate}
```

Penjelasan khusus tentang penulisan *optional argument* sebagaimana `[({A}-1)]` di contoh terakhir adalah sebagai berikut:

1. Teks yang ditulis di dalam kurung kurawal adalah teks statis (tidak berubah sesuai urutan) yang akan ada di setiap *item*. Penulisan {A} akan menghasilkan luaran A di setiap item, tidak berubah menjadi B, C, dan seterusnya.

2. Teks yang ditulis tanpa kurung kurawal juga menjadi teks statis **asalkan** di dalam teks itu tidak ada salah satu dari karakter `1, a, A, i,` atau `I`. Dari contoh terakhir, karakter `(, -,` dan `)` terbaca sebagai teks statis.

3. Salah satu dari karakter `1, a, A, i,` atau `I` yang ditulis tanpa kurung kurawal akan berubah sesuai urutan (baik itu urutan nomor, abjad, ataupun angka Romawi) di setiap *item*. Jika karakter-karakter ini ditemukan lebih dari satu dan sama-sama ditulis tanpa kurung kurawal, karakter yang akan dimunculkan sesuai urutan pada luarannya adalah karakter yang terakhir. Karakter yang lain akan terbaca sama dengan karakter yang terakhir. Misal kita menuliskan *optional argument* `[A-1]`, luarannya menjadi 1-1, 2-2, 3-3, dan seterusnya.

4. Di dalam *optional argument* ini juga dikenal (dapat dituliskan) spasi, termasuk perintah \hspace{lebar spasi}.

2.13 Catatan Kaki

Cara membuat catatan kaki adalah dengan menggunakan perintah `\footnote{Teks catatan kaki yang ingin dituliskan}`. Perintah ini ditulis di bagian teks yang di sana ingin diberi catatan kaki.

Contoh:
Lihat catatan kaki[1] di bagian bawah halaman ini! Perintah untuk membuatnya adalah

```
\footnote{Ini adalah contoh catatan kaki.  Penomoran
untuk catatan kaki ini otomatis.}.
```

2.14 Teks Multi Kolom

Banyak jenis dokumen yang aturan penulisannya mensyaratkan agar teks dibuat lebih dari satu kolom. Salah satunya adalah untuk keperluan penulisan artikel yang akan disubmit ke suatu jurnal ilmiah. Teks dua kolom dapat dibuat dengan menuliskan `twocolumn` pada *optional argument* saat menuliskan kelas dokumen (perintah paling awal). Akan tetapi pengaturan teks menjadi dua kolom dengan cara seperti ini akan berlaku untuk seluruh teks di dalam dokumen.

Untuk membuat teks multi kolom (bisa lebih dari dua kolom) dan berlaku tidak harus untuk seluruh teks di dokumen, kita perlu tambahan *package* `multicol`. Pada bagian *preamble* tuliskan `\usepackage{multicol}`! Selanjutnya, untuk membuat teks menjadi `n` kolom, digunakan *environment* berikut:

```
\begin{multicols}{n}
Teks yang ingin dibuat menjadi n kolom
\end{multicols}
```

Ganti `n` dengan bilangan sesuai banyaknya kolom yang diinginkan!

Contoh:

[1] Ini adalah contoh catatan kaki. Penomoran untuk catatan kaki ini otomatis.

Perhatikan teks dua kolom berikut!

David Hilbert, seorang matematikawan Jerman, mendaftar dua puluh tiga masalah matematika yang pada masa itu belum terpecahkan. Masalah-masalah ini lebih dikenal sebagai Masalah-masalah Hilbert (*Hilbert's Problems*). Daftar masalah ini dikemukakan oleh Hilbert pada Kongres Matematikawan Internasional tanggal 8 Agustus 1900 di Paris, Perancis.

Teks di atas ditulis dengan *environment*

```
\begin{multicols}{2}
David Hilbert, seorang matematikawan Jerman, mendaftar
dua puluh tiga masalah matematika yang pada masa itu
belum terpecahkan.  Masalah-masalah ini lebih dikenal
sebagai Masalah-masalah Hilbert (\textit{Hilbert's
Problems}).  Daftar masalah ini dikemukakan oleh Hilbert
pada Kongres Matematikawan Internasional tanggal 8
Agustus 1900 di Paris, Perancis.
\end{multicols}
```

2.15 Mewarnai Teks

Untuk keperluan pewarnaan teks, kita bisa menggunakan *package* `xcolor`. Tuliskan di *preamble* `\usepackage{xcolor}`! Selanjutnya untuk mewarnai teks, perintahnya adalah `\color{warna}` (ganti `warna` sesuai dengan warna yang diinginkan!). Warna seluruh teks yang ditulis setelah perintah ini akan berubah sesuai pengaturan. Untuk membatasi agar hanya teks tertentu yang diwarnai, gunakan kurung kurawal untuk mengapit perintah beserta teks yang diwarnai! Cara lain juga dapat dilakukan, yakni dengan perintah `\textcolor{warna}{teks yang diwarnai}`.

Contoh:
Teks berwarna berikut ini,

Ini teks warna merah dan ini teks warna biru

dapat ditulis dengan kode

```
{\color{red}Ini teks warna merah }\textcolor{blue}{dan
ini teks warna biru}
```

Daftar `warna` yang dapat dipilih untuk digunakan di antaranya adalah `black`, `blue`, `brown`, `darkgray`, `gray`, `lightgray`, `red`, `purple`, `pink`, `lime`, `yellow`, `green`, `magenta`, `violet`, `olive`, `teal`, `orange`, dan `white`. Kita juga dapat membuat warna n persen kali lebih cerah dengan menambahkan `!n` tepat setelah penulisan `warna`. Lihat perbedaan warna `purple!50`, `purple!75`, dan `purple`.

Selain pada teksnya, pewarnaan juga bisa diberikan pada *background* teks. Perintahnya adalah `\colorbox {warna}{teks yang diberi background berwarna}`. Jika *background* teks selain diwarnai juga ingin diberi garis tepi berwarna, perintahnya adalah `\fcolorbox{warna garis tepi}{warna background}{teks yang diberi background bergaris}`.

Contoh:
Teks berikut ini,

Di sini ada banyak warna

dapat ditulis menggunakan kode berikut:

```
\colorbox{teal}{\color{white}Di sini } \fcolorbox{blue}
{lime!40}{\color{purple}ada banyak warna}
```

2.16 Mengatur Penomoran Halaman

Ada beberapa *style* penomoran halaman. Tanpa pengaturan khusus, *style* penomoran halaman yang digunakan di berbagai dokumen LaTeX adalah angka biasa. Macam-macam *style* penomoran dan kodenya yaitu meliputi: angka biasa (`arabic`), huruf kecil (`alph`), huruf kapital (`Alph`), angka romawi kecil (`roman`), dan angka romawi kapital (`Roman`). Cara menggunakan *style* penomoran dalam dokumen kita (untuk seluruh halaman dalam dokumen) adalah dengan

menuliskan perintah \pagenumbering{kode style} di *preamble*.

Untuk kelas dokumen book, dua *style* penomoran halaman biasanya digunakan sekaligus dalam satu dokumen. Dua *style* itu adalah angka romawi kecil (digunakan sebelum mulai bab pertama) dan angka biasa (digunakan dari awal bab pertama). Untuk penomoran halaman buku, LaTeX telah memberikan cara mudah, yaitu dengan mengelompokkan halaman-halaman ke dalam dua jenis, meliputi halaman-halaman depan dan halaman-halaman inti. Perintah untuk pengelompokan halaman-halaman depan adalah \frontmatter (biasa ditulis tepat setelah \begin{document}) dan perintah untuk pengelompokan halaman-halaman inti adalah \mainmatter (biasa ditulis sebelum perintah \chapter yang pertama). Dengan dua perintah ini saja, tanpa perintah \pagenumbering, penomoran halaman sudah menjadi dua *style*.

Untuk mengubah desain penomoran, dibutuhkan tambahan *package* tertentu. Subbab ini tidak menjelaskan cara mengubah desain penomoran. Silakan belajar mandiri dari sumber lain jika tertarik.

2.17 Membuat Daftar Isi

Daftar isi secara otomatis dapat dibuat dengan menuliskan perintah \tableofcontents. Untuk melihat hasil, kode yang sudah dituliskan harus ***dicompile* dua kali**. Hasilnya bisa dimulai dengan tulisan "Table of Contents" atau "Daftar Isi", bergantung pada bahasa yang digunakan. Jika menginginkan hasilnya berbahasa Indonesia, pastikan di bagian *preamble* telah menuliskan \usepackage[bahasa]{babel}.

Ada kalanya terdapat suatu halaman yang kita inginkan terdaftar atau disebut juga di daftar isi tetapi ternyata tidak muncul di daftar isi karena tidak dikenali sebagai bab, subbab, atau yang lain. Untuk kasus seperti ini, pada bagian halaman itu tuliskan perintah \addcontentsline{toc}{...}{teks}! Ganti (isi) titik-titik dengan chapter, section, atau yang lain, sesuai dengan ingin sebagai apa ia dikenali! Ganti teks dengan teks (nama judul atau yang lain) yang ingin dimunculkan di daftar isi.

Di dalam *list* daftar isi, biasanya juga tidak ada kata-kata "Daftar Isi" berikut nomor halamannya. Khusus untuk hal ini, jika

menginginkan "Daftar Isi" muncul di dalam *list*, tambahkan saja \usepackage{tocbibind} di *preamble*!

Tidak ada contoh untuk subbab ini, tetapi pembaca dapat melihat daftar isi buku ini. Daftar isi buku ini dibuat dengan menuliskan \tableofcontents dan menambahkan *package* tocbibind.

2.18 Membuat Hyperlink

Dokumen yang dihasilkan oleh LaTeX, yakni berupa file pdf, di dalamnya dapat memuat hyperlink-hyperlink. Hyperlink dapat dibuat dengan menambahkan *package* hyperref. Dengan hanya menuliskan \usepackage{hyperref} di *preamble*, daftar isi akan menjadi hyperlink-hyperlink, yakni bisa diklik untuk "melompat" ke halaman-halaman tertentu. Demikian juga dengan sitasi, akan menjadi hyperlink yang mengantarkan kita ke daftar pustaka.

Kita juga dapat membuat hyperlink berupa url. Ada dua macam hyperlink dalam hal ini, yaitu:

1. **url tampil secara langsung sebagai teks**. Perintah yang digunakan adalah \url{url yang dituju}. Ganti url yang dituju dengan suatu url atau alamat website! Luaran yang dihasilkan dari perintah ini adalah teks berupa url yang tampil di dokumen dan dapat diklik untuk membuka url itu.

2. **url tidak tampil sebagai teks**. Perintah yang harus dituliskan adalah \href{url yang dituju}{deskripsi}. Ganti url yang dituju dengan suatu url dan ganti deskripsi dengan teks yang akan dimunculkan di dokumen! Luaran yang dihasilkan dari perintah ini adalah teks (sesuai deskripsi) dan berfungsi sebagai hyperlink yang dapat diklik untuk membuka url itu.

Contoh:

1. Jika kita menulis kode \url{http://irmansaid.uinws.id/} maka yang muncul di dokumen adalah http://irmansaid.uinws.id/ (sebagai hyperlink).

2. Jika kita menulis kode \href{http://irmansaid.uinws.id/}{blogku} maka yang muncul di dokumen adalah teks blogku (sebagai hyperlink).

Tanpa pengaturan khusus, hyperlink biasa dimunculkan di teks dengan kotak berwarna yang menandakan batas wilayah yang dapat diklik. Bagi sebagian orang ini mengganggu tampilan meskipun jika dokumen dicetak (diprint), kotak ini tidak muncul di hasil cetak. Untuk menghilangkan kotak berwarna itu caranya adalah dengan menyertakan *optional argument* `hidelinks` pada pemanggilan *package* `hyperref` di *preamble*, yakni perintahnya menjadi `\usepackage[hidelinks]{hyperref}`. Tanpa pengaturan warna tertentu, `hidelinks` juga membuat warna hyperlink sama dengan warna teks asli.

Warna hyperlink juga dapat diatur sesuai keinginan dengan menambahkan beberapa kode (masing-masing dipisahkan oleh koma) di *optional argument* pada pemanggilan *package* `hyperref`. Kode yang pertama adalah `colorlinks=true` (jika diganti `false` maka pewarnaan tidak bisa dilakukan). Berikutnya untuk mengatur warna, kode-kodenya adalah sebagai berikut: `linkcolor` (mengatur warna link internal), `citecolor` (mengatur warna sitasi), `urlcolor` (mengatur warna link untuk url), dan `allcolors` (mewarnai semua link dengan satu warna yang sama). Cara menambahkan kode ini di *optional argument* adalah dengan diikuti `=warna` (ganti `warna` sesuai keinginan!).

Contoh:
Untuk menghilangkan kotak pada hyperlink dan membuat semua link berwarna biru, perintah pada *preamble* menjadi

```
\usepackage[hidelinks,colorlinks=true,allcolors=blue]
{hyperref}
```

Bab 3

Pengetikan Teks Matematis

Salah satu keunggulan LaTeX adalah bisa menghasilkan penulisan rumus dan karakter-karakter matematis yang rapi. Ini pula yang banyak menjadi alasan orang fisika atau matematika memilih menulis dokumen menggunakan LaTeX. Untuk dapat menuliskan rumus-rumus matematis, syarat awal adalah di bagian *preamble* harus menuliskan *package* untuk pengetikan matematis, misal:

```
\usepackage{amsmath}
```

Ini adalah *package* yang paling banyak digunakan meskipun ada *package* lain dengan kegunaan serupa.

3.1 Mode *Inline* dan *Displaymath*

Dalam pengetikan dokumen menggunakan LaTeX, teks matematis terbedakan dengan teks biasa. Teks matematis sendiri terbedakan atas dua macam tampilan: mode *inline* (teks matematis ditulis dalam baris yang sama dengan teks biasa) dan mode *displaymath* (teks matematis ditampilkan secara khusus dan terpisah dari baris

teks biasa).

Contoh penggunaan mode *inline*:
Definisi turunan diberikan oleh $\frac{df}{dx} := \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}$.

Contoh penggunaan mode *displaymath*:
Definisi turunan diberikan oleh

$$\frac{df}{dx} := \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}.$$

Untuk membuat teks matematis diperlukan perintah khusus sesuai tampilan yang diinginkan. Perintah untuk membuat mode *inline* ada beberapa cara, yakni (titik-titik adalah tempat menulis teks matematis):

1. `\(...)\`
2. `$...$`
3. `\begin{math}...\end{math}`

Perintah untuk membuat mode *displaymath* ada beberapa cara, yakni:

1. `\begin{displaymath}...\end{displaymath}`
2. `\[...\]`
3. `$$...$$`
4. `\begin{equation}...\end{equation}`
   Cara keempat hasilnya disertai dengan nomor persamaan. Nomor persamaan bisa tidak ikut disertakan dengan penambahan *, yakni `\begin{equation*} ...\end{equation*}`.

Penting untuk diingat, **seluruh teks matematis yang nanti dituliskan harus berada di salah satu dari mode *inline* atau *displaymath***. Artinya, tidak bisa kita menuliskan pecahan, huruf Yunani, atau yang lain, tanpa menuliskan (katakanlah minimal yang paling sering digunakan adalah) `$...$`. Jika kita "nekat" tidak menuliskannya, dokumen akan *error* saat dicompile.

3.2 Spasi

Spasi horisontal dari yang sempit sampai yang lebar pada teks matematis dibuat dengan perintah-perintah berikut: `\,` , `\:` , `\;` , `\quad`, dan `\qquad`. Perintah `\hspace{...}` juga masih dapat digunakan di sini.

Contoh:
1 2 3 4 5 6 7
dibuat dengan kode
`$1\,2\:3\;4\quad 5\qquad 6\hspace{1cm}7$`.

3.3 Titik-titik

Ada perintah khusus untuk membuat titik tiga sekaligus, yaitu `\ldots` (di bawah mendatar $\ldots$), `\cdots` (di tengah mendatar $\cdots$), `\vdots` (arah vertikal $\vdots$) dan `\ddots` (arah diagonal $\ddots$). Ada pula perintah untuk membuat satu titik di tengah ($\cdot$), yaitu `\cdot`.

Contoh:

1. Barisan $1, 2, 3, \ldots, 9$ dibuat menggunakan kode
   `$1,2,3,\ldots,9$`.
2. $2(5 + 7) = 2 \cdot 5 + 2 \cdot 7$ dibuat dengan kode
   `$2(5+7)=2\cdot 5+2\cdot 7$`.

3.4 *Subscript* dan *Superscript*

Perintah untuk membuat *subscript* adalah `_{...}` dan untuk membuat *superscript* adalah `^{...}`. Jika hanya satu karakter yang ingin dibuat *subscript* atau *superscript*, kurung kurawal tidak harus digunakan.

Contoh:

1. a^b dibuat dengan perintah `$a^b$`.

2. $(x_n)_{n=1}^{100}$ dibuat dengan menuliskan `$(x_n)_{n=1}^{100}$`.

3. ${}^nC_r$ dibuat dengan perintah `$^nC_r$`.

3.5 Pecahan

Perintah untuk menulis pecahan adalah `\frac{...}{...}` (ukuran karakter mengecil di mode *inline*) atau `\dfrac{...}{...}`.

Contoh:
Perhatikan perbedaan penulisan pecahan $\frac{22}{7}$ dengan $\dfrac{22}{7}$! Keduanya ditulis berturut-turut dengan perintah
`$\frac{22}{7}$`
dan
`$\dfrac{22}{7}$`.

3.6 Akar

Perintah untuk menulis akar adalah `\sqrt{...}`. Jika kita ingin menulis akar pangkat n (suatu bilangan) maka diperlukan tambahan *optional argument* yang berisi bilangan n tersebut, yakni menjadi `\sqrt[n]{...}`.

Contoh:

1. $\sqrt{2}$ dibuat dengan perintah `$\sqrt{2}$`

2. $$\sqrt[3]{\frac{5\sqrt{2}}{7}}$$

   dibuat dengan perintah
   `\[\sqrt[3]{\frac{5\sqrt{2}}{7}}\]`

3.7 Karakter-karakter Khusus

Berbagai karakter matematis ditulis menggunakan perintah-perintah khusus. Tabel 3.1 mendaftar perintah untuk masing-masing karakter itu tetapi karakter-karakter yang terdaftar di tabel terbatas pada karakter-karakter yang (menurut penulis) sering digunakan.

Tabel 3.1: Daftar perintah untuk menuliskan karakter-karakter khusus

Perintah	Luaran
`\forall`	$\forall$
`\exists`	$\exists$
`\emptyset`	$\emptyset$
`\wedge`	$\wedge$
`\vee`	$\vee$
`\neg`	$\neg$
`\in`	$\in$
`\ni`	$\ni$
`\notin`	$\notin$
`\subset`	$\subset$
`\supset`	$\supset$
`\subseteq`	$\subseteq$
`\supseteq`	$\supseteq$
`\cap`	$\cap$
`\cup`	$\cup$
`\dagger`	$\dagger$
`\star`	$\star$
`\perp`	$\perp$
`\angle`	$\angle$
`\nabla`	∇
`\partial`	∂
`\hbar`	$\hbar$
`\infty`	∞
`\propto`	$\propto$
`\sim`	$\sim$

Perintah	Luaran
\approx	$\approx$
\simeq	$\simeq$
\cong	$\cong$
\equiv	$\equiv$
\neq	$\neq$
\leq	$\leq$
\geq	$\geq$
\ll	$\ll$
\gg	$\gg$
\times	$\times$
\div	$\div$
\circ	$\circ$
\pm	$\pm$
\mp	$\mp$
\oplus	$\oplus$
\otimes	$\otimes$
\odot	$\odot$
\int	$\int$
\oint	$\oint$
\iint	$\iint$
\iiint	$\iiint$
\bigcap	$\bigcap$
\bigcup	$\bigcup$
\sum	$\sum$
\prod	$\prod$
\coprod	$\coprod$
\langle	$\langle$
\rangle	$\rangle$
\\|	$\|$

Contoh:

1. $\forall x \in A, x \in A \cup \emptyset$ ditulis dengan kode
   `$\forall x\in A,x\in A\cup\emptyset$`

2. $|\nabla \times \nabla f| = 0$ ditulis dengan kode
   `$|\nabla\times\nabla f|=0$`
3. $(a \div b) + c \neq a \div (b + c)$ ditulis dengan kode
   `$(a\div b)+c\neq a\div (b+c)$`

3.8 Huruf-huruf Yunani

Huruf-huruf Yunani dituliskan dengan perintah-perintah khusus sebagaimana yang terdaftar di tabel 3.2.

Tabel 3.2: Daftar perintah untuk menuliskan huruf-huruf Yunani

Perintah	Luaran
`\alpha`	α
`\beta`	β
`\gamma`	γ
`\delta`	δ
`\epsilon`	ϵ
`\varepsilon`	ε
`\zeta`	ζ
`\eta`	η
`\theta`	θ
`\vartheta`	ϑ
`\iota`	ι
`\kappa`	κ
`\lambda`	λ
`\mu`	μ
`\nu`	ν
`\xi`	ξ
`\pi`	π
`\varpi`	ϖ
`\rho`	ρ
`\varrho`	ϱ
`\sigma`	σ
`\varsigma`	ς

Perintah	Luaran
\tau	τ
\upsilon	υ
\phi	ϕ
\varphi	φ
\chi	χ
\psi	ψ
\omega	ω
\Gamma	Γ
\Delta	Δ
\Theta	Θ
\Lambda	Λ
\Xi	Ξ
\Pi	Π
\Sigma	Σ
\Upsilon	Υ
\Phi	Φ
\Psi	Ψ
\Omega	Ω

Contoh:
Ungkapan $dV = \rho d\rho d\varphi dz$ dihasilkan dari penulisan kode berikut:
`$dV=\rho d\rho d\varphi dz$`

3.9 Karakter-karakter Khusus dengan Batas

Beberapa karakter khusus dan huruf Yunani dalam matematika digunakan sebagai notasi singkat untuk penjumlahan n suku, perkalian n suku, penggabungan atau irisan n buah himpunan, dan lain-lain, yang dalam penulisannya melibatkan batas bawah dan batas atas. Termasuk di dalamnya adalah notasi Sigma untuk penjumlahan, notasi integral, dan masih banyak yang lainnya.

Untuk menuliskan batas-batas yang menyertai karakter itu, se-

telah perintah penulisan karakter, digunakan perintah pembuatan *subscript* dan *superscript*. Untuk hasil yang berbeda ("lebih baik") dalam mode *inline*, tambahkan perintah `\limits` sebelum perintah *subscript* dan *superscript*.

Contoh:

1. Ungkapan $M = \sum_{i=1}^{N} m_i$ dan $M = \sum\limits_{i=1}^{N} m_i$ (keduanya dalam mode *inline*, perhatikan letak penulisan batas-batasnya!) berturut-turut ditulis dengan perintah
   `$M=\sum_{i=1}^N m_i$`
   dan
   `$M=\sum\limits_{i=1}^N m_i$`

2. Ungkapan
   $$\Gamma(\alpha) := \int_0^{\infty} e^{-x} x^{\alpha-1} dx$$
   ditulis dengan perintah
   ```
   \[
   \Gamma(\alpha):=\int_0^{\infty}e^{-x}x^{\alpha-1}dx
   \]
   ```

3. Ungkapan
   $$ds^2 = \sum_{\mu,\nu=0}^{3} g_{\mu\nu} dx^{\mu} dx^{\nu}$$
   ditulis dengan perintah
   ```
   \[
   ds^2=\sum_{\mu,\nu=0}^3g_{\mu,\nu}dx^{\mu}dx^{\nu}
   \]
   ```

3.10 Anak Panah

Ada banyak macam jenis anak panah dan tidak semuanya akan disebutkan di sini. Beberapa di antara perintah-perintah untuk membuat beragam jenis anak panah itu adalah seperti terdaftar pada tabel 3.3.

Tabel 3.3: Daftar perintah untuk membuat anak-anak panah

Perintah	Luaran
\to	$\to$
\leftarrow	$\leftarrow$
\uparrow	$\uparrow$
\rightarrow	$\rightarrow$
\downarrow	$\downarrow$
\updownarrow	$\updownarrow$
\leftrightarrow	$\leftrightarrow$
\Leftarrow	$\Leftarrow$
\Uparrow	$\Uparrow$
\Rightarrow	$\Rightarrow$
\Downarrow	$\Downarrow$
\Updownarrow	$\Updownarrow$
\Leftrightarrow	$\Leftrightarrow$
\mapsto	$\mapsto$
\nearrow	$\nearrow$
\searrow	$\searrow$
\swarrow	$\swarrow$
nwarrow	$\nwarrow$
\leftharpoonup	$\leftharpoonup$
\rightharpoonup	$\rightharpoonup$
\leftharpoondown	$\leftharpoondown$
\rightharpoondown	$\rightharpoondown$

Contoh:
Pemetaan $f : X \to Y : x \mapsto f(x)$ dibuat dengan perintah
`$f:X\to Y:x\mapsto f(x)$`

3.11 Fungsi-fungsi Matematis

Beberapa fungsi matematis yang umum dikenal seperti trigonometri, logaritma, eksponensial, dan beberapa yang lain, cara penulisan

standarnya adalah menggunakan huruf tegak, bukun huruf miring. Ini berbeda dengan penulisan huruf biasa sebagai karakter matematis yang biasa ditulis miring. Untuk membuat tulisannya menjadi tegak, disediakan perintah-perintah khusus, yakni *backslash* (\) disertai nama fungsinya.

Perintah-perintah itu meliputi: `\lim`, `\limsup`, `\liminf`, `\lg`, `\ker`, `\sin`, `\cos`, `\tan`,`\csc`, `\sec`, `\cot`, `\arcsin`,`\arccos`, `\arctan`, `\sinh`, `\cosh`, `\tanh`, `\coth`, `\inf`, `\sup`, `\exp`, `\ln`, `\log`, `\det`, `\min`, `\max`, `\dim`, `\deg`, `\gcd` dan `\hom`.

Contoh:
Ungkapan

$$\ln x = {}^{e}\log x$$

ditulis menggunakan perintah
`\[\ln x=^e\log x\]`

Bandingkan penulisan `$\ln x$` dan `$ln x$` (secara langsung tanpa \)! Luaran yang dihasilkan berbeda, yakni $\ln x$ dan lnx. Penulisan yang benar adalah yang pertama. Ini sebenarnya hanya tentang kesepakatan dan orang bisa saja membuat kesepakatan yang lain.

3.12 Vektor

Tanda anak panah atas pada penulisan vektor diberikan oleh perintah `\vec{...}` dan tanda topi untuk vektor satuan diberikan oleh perintah `\hat{...}`.

Contoh:
Perintah untuk menuliskan vektor $\vec{V} = V_x\hat{i} + V_y\hat{j}$ adalah
`$\vec{V}=V_x\hat{i}+V_y\hat{j}$`

3.13 Aksen-aksen Matematis

Tanda anak panah atas dan topi pada vektor sebenarnya termasuk ke dalam aksen-aksen matematis meskipun pembahasannya senga-

ja dipisahkan di subbab sebelumnya. Kita akan membahas aksen-aksen matematis yang lain. Untuk membuat garis di atas suatu karakter (bar), tilde, titik atas, titik dua atas, dan titik tiga atas, perintah yang digunakan berturut-turut adalah `\bar{...}`, `\tilde{...}`, `\dot{...}`, `\ddot{...}`, dan `\dddot{...}` (titik-titik diganti dengan suatu karakter).

Contoh:

1. Perintah untuk menuliskan $a = \dot{v} = \ddot{x}$ adalah
   `$a=\dot{v}=\ddot{x}$`

2. Perintah untuk menulis

$$\bar{\tilde{x}} = \frac{\tilde{x}_1 + \tilde{x}_2}{2}$$

   adalah
   `\[\bar{\tilde{x}}=\frac{\tilde{x}_1+\tilde{x}_2}{2}\]`

Kita juga dapat membuat garis atas panjang sepanjang karakter-karakter yang ada di bawahnya dengan perintah `\overline{...}`, kurung kurawal mendatar panjang di atas dan di bawah karakter dengan perintah `\overbrace{...}` dan `\underbrace{...}`, anak panah panjang di atas (semacam notasi vektor) mengarah ke kanan dan mengarah ke kiri dengan perintah `\overrightarrow{...}` dan `\overleftarrow{...}`, serta topi panjang di atas dengan perintah `\widehat{...}`.

Contoh:

1. Persamaan $\overline{x + iy} = x - iy$ ditulis dengan kode
   `$\overline{x+iy}=x-iy$`

2. Barisan

$$\underbrace{1, 1, 1, \ldots, 1}_{n}, \overbrace{-1, -1, -1, \ldots, -1}^{m}$$

ditulis menggunakan kode
```
\[\underbrace{1,1,1,\ldots,1}_n,
\overbrace{-1,-1,-1,\ldots,-1}^m\]
```

3. $\overrightarrow{AB} = \widehat{PQ}$ ditulis dengan kode
```
$\overrightarrow{AB}=\widehat{PQ}$
```

3.14 Matriks

Matriks dapat dibuat menggunakan *environment* `matrix, pmatrix, bmatrix,` atau `Bmatrix` (akan menghasilkan tampilan matriks yang berbeda). Untuk menuliskan determinan matriks dengan garis tegak kiri dan kanan, digunakan *environment* `vmatrix`. Jika garis tegak ingin dibuat rangkap dua, digunakan *environment* `Vmatrix`. Di dalam *environment*, perintah untuk berpindah kolom menggunakan `&` dan untuk berpindah baris menggunakan `\\`.

Contoh:

1. Matriks $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ dibuat dengan *environment* berikut:

```
$
\begin{pmatrix}
1&2&3\\
4&5&6\\
7&8&9
\end{pmatrix}
$
```

2. Matriks $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ dibuat dengan *environment* berikut:

```
$
\begin{bmatrix}
a&b&c\\
```

```
d&e&f\\
g&h&i
\end{bmatrix}
$
```

3.15 Kurung Berketinggian Sesuai Karakter

Yang dimaksud kurung berketinggian sesuai karakter adalah kurung yang ketinggiannya menyesuaikan tinggi teks matematis yang diapit. Kurung semacam ini ditulis dengan perintah pembuka dan perintah penutup (**harus ada dua-duanya** meskipun jenis kurung tidak sama kiri dengan kanan). Daftar perintah pembuka dan penutup ada pada tabel 3.4.

Tabel 3.4: Daftar perintah untuk membuat kurung setinggi teks matematis yang diapit

Perintah Pembuka	Luaran	Perintah Penutup	Luaran
`\left.`		`\right.`	
`\left|`	$\left\vert\right.$	`\right|`	$\left.\right\vert$
`\left\\|`	$\left\Vert\right.$	`\right\\|`	$\left.\right\Vert$
`\left(`	$\left($	`\right)`	$\left.\right)$
`\left[`	$\left[$	`\right]`	$\left.\right]$
`\left\{`	$\left\{$	`\right\}`	$\left.\right\}$
`\left\langle`	$\left\langle$	`\right\rangle`	$\left.\right\rangle$
`\left)`	$\left)$	`\right(`	$\left.\right($
`\left]`	$\left]$	`\right[`	$\left.\right[$
`\left\}`	$\left\}$	`\right\{`	$\left.\right\{$
`\left\rangle`	$\left\rangle$	`\right\langle`	$\left.\right\langle$

Khusus untuk perintah `\left.` dan `\right.`, keduanya tidak akan menghasilkan kurung, tetapi harus ada sebagai pelengkap jika kita ingin membuat kurung satu sisi saja.

Contoh:

1. Untuk menuliskan
$$\left|\frac{1}{2}\right\rangle$$
dilakukan dengan mengetikkan
`\[\left|\frac{1}{2}\right\rangle\]`.

2. Untuk menuliskan
$$\left.\frac{df}{dx}\right|_{x=c}$$
dilakukan dengan mengetikkan
`\[\left.\frac{df}{dx}\right|_{x=c}\]`.

3.16 Teks Matematis Beberapa Baris

Teks matematis beberapa baris dapat dibuat dengan *environment* `align`. Tiap baris memiliki nomor persamaan, kecuali jika pada penulisan `align` diberi tanda `*`, yakni menjadi `align*`. Jika ingin menghilangkan nomor persamaan di baris tertentu saja maka pada baris itu dapat diberi perintah `\nonumber`. Selanjutnya untuk meratakan teks pada bagian tertentu antar baris (misal meratakan =), digunakan `&` dan untuk berpindah baris digunakan `\\`.

Contoh:
Persamaan-persamaan berikut ini,

$$\begin{aligned} {}^5P_2 &= \frac{5!}{(5-2)!} && (3.1)\\ &= \frac{5\cdot 4\cdot 3!}{3!} \\ &= 20 && (3.2)\end{aligned}$$

dibuat dengan perintah:

```
\begin{align}
^5P_2&=\frac{5!}{(5-2)!}\\
&=\frac{5\cdot 4\cdot 3!}{3!}\nonumber\\
```

```
&=20
\end{align}
```

Environment `align` di sini setara dengan *environment* `equation` dan semacamnya tetapi dengan kekhususan perintah untuk membuat banyak baris. Artinya, *environment* `align` tidak di dalam *environment* `equation` atau yang sejenisnya, seperti `$...$`.

Ada cara lain yang dapat ditempuh untuk membuat teks matematis beberapa baris dengan perataan di bagian-bagian tertentu, yaitu menggunakan *environment* `array`. Dengan *environment* `array`, penggunaan `&` (untuk meratakan teks ke bawah) dalam tiap baris tidak terbatas hanya satu tetapi bisa beberapa. Cara pembuatan *environment* `array` berikut isinya sangat dekat dengan cara pembuatan tabel. Dengan alasan ini, pembahasan tentang *environment* `array` akan di berikan di belakang pada subbab 5.9, setelah pembahasan tabel.

3.17 Pelabelan Persamaan

Setiap persamaan matematis dapat dilabeli. Pemberian label dapat dilakukan dengan menuliskan perintah `\label{Nama label}`. Perintah ini ditulis di baris setelah persamaan atau di sembarang tempat di dalam *environment* tempat ditulisnya persamaan itu jika *environment* tersebut hanya dapat memunculkan satu nomor persamaan, misal *environment* `equation`. Untuk *environment* `align` (atau yang lain) yang dapat memunculkan banyak nomor persamaan, label persamaan dituliskan setelah persamaan. Untuk amannya, **tuliskan saja label setelah persamaan!**

Kegunaan pemberian label adalah agar persamaan dapat diacu atau disebut dari tempat lain. Cara menyebut persamaan yang dilabeli dari tempat lain adalah dengan menuliskan perintah `\ref{Nama label}`. Untuk melihat hasilnya, lakukan ***compile*** **dua kali!**.

Contoh:
Perhatikan teks berikut persamaan-persamaannya di bawah ini!

Pada benda yang bergerak vertikal di bawah pengaruh medan gra-

vitasi bumi, berlaku persamaan

$$v(t) = v_0 - gt. \tag{3.3}$$

Untuk kasus gerak jatuh bebas, berlaku

$$v_0 = 0. \tag{3.4}$$

Substitusi persamaan (3.4) ke persamaan (3.3) menghasilkan persamaan

$$v(t) = -gt. \tag{3.5}$$

Kode yang digunakan untuk menuliskan teks di atas adalah

```
Pada benda yang bergerak vertikal di bawah pengaruh
medan gravitasi bumi, berlaku persamaan
\begin{equation}\label{abc}
v(t)=v_0-gt.
\end{equation}
Untuk kasus gerak jatuh bebas, berlaku
\begin{equation}\label{def}
v_0=0.
\end{equation}
Substitusi persamaan (\ref{def}) ke persamaan (\ref{abc})
menghasilkan persamaan
\begin{equation}
v(t)=-gt.
\end{equation}
```

Sebagai tambahan informasi, pemberian label secara umum tidak terbatas hanya dapat dilakukan pada persamaan matematis tetapi juga dapat digunakan lebih luas pada **bab, subbab, gambar, tabel, dan sebagainya** sehingga nomor bab, subbab, gambar, tabel, dan yang lainnya itu dapat diacu atau disebut dari tempat yang lain.

3.18 Teks Matematis Tebal dan Miring

Untuk membuat teks matematis bercetak tebal dan bercetak miring, perintahnya berturut-turut yaitu `\mathbf{...}` dan `\mathit{...}`. Titik-titik diisi teks matematis yang ingin dibuat bercetak tebal atau miring.

Contoh:
Persamaan berikut ini,

$$Gaya = \mathbf{F} = \frac{d\mathbf{p}}{dt}$$

dapat ditulis menggunakan kode

```
\[\mathit{Gaya}=\mathbf{F}=\frac{d\mathbf{p}}{dt}\]
```

Seandainya `\nathit` tidak dituliskan pada contoh ini hasilnya tetap sama karena standar huruf sebagai karakter matematis memang sudah bercetak miring.

Untuk tambahan, perintah membuat teks bergaris bawah masih tetap sama baik itu di teks biasa ataupun teks matematis, yaitu `\underline{...}`.

3.19 Ubah Font Matematis

Jenis font matematis untuk karakter biasa (abjad) dapat diubah dengan cara dikenai salah satu dari beberapa perintah berikut: `\mathrm{...}`, `\mathsf{...}`, `\mathtt{...}`, `\mathbb{...}`, `\mathcal{...}`, atau `\mathfrak{...}` (untuk jenis font yang berbeda-beda). Akan tetapi untuk `mathbb` dan `mathcal` harus dikenakan pada tulisan dengan huruf kapital.

Contoh:

1. Perhatikan font dari huruf R yang digunakan untuk menyatakan himpunan bilangan riil pada pernyataan matematis berikut!

$$(1, 2) \in \mathbb{R}^2$$

Kita dapat menuliskan pernyataan di atas menggunakan kode
`\[(1,2)\in\mathbb{R}^2\]`

2. Notasi dari sebuah aljabar Lie kompleks berikut ini,

$$\mathfrak{sl}_n\mathbb{C}$$

ditulis menggunakan kode
`\[\mathfrak{sl}_n\mathbb{C}\]`

3.20 Teks Biasa di Dalam Teks Matematis

Teks biasa dapat ditulis di dalam teks matematis dengan terlebih dulu menuliskan perintah `\text{Ini jadi teks biasa}`.

Contoh:
Untuk menuliskan

$$|a| = 1 \text{ jika dan hanya jika } a = 1 \text{ atau } a = -1$$

dapat digunakan kode berikut:

```
\[
|a|=1\text{ jika dan hanya jika }a=1\text{ atau }a=-1
\]
```

Bandingkan dengan apabila kita mengetikkan kode berikut:

```
\[
|a|=1 jika dan hanya jika a=1 atau a=-1
\]
```

Kode di atas hasilnya berantakan seperti di bawah ini:

$$|a| = 1jikadanhanyajikaa = 1ataua = -1$$

Bahkan, spasi yang kita buat (dengan menekan tombol spasi pada *keyboard*) tidak dikenal sebagai spasi. Dalam teks matematis ada cara tersendiri untuk pembuatan spasi sebagaimana yang telah dijelaskan sebelumnya. Di samping itu, seluruh huruf juga ditulis bercetak miring karena standar huruf-huruf sebagai karakter-karakter matematis memang dibuat miring, berbeda dengan teks biasa.

Bab 4

Menambahkan File Gambar

Untuk memasukkan file gambar ke dokumen ataupun membuat gambar secara langsung menggunakan perintah-perintah (kode) di LaTeX, diperlukan penggunaan *package* tertentu. Dari beberapa *package* yang mungkin digunakan, *package* yang paling sering dipilih adalah `graphicx`. Di bagian *preamble* perlu dituliskan

```
\usepackage{graphicx}
```

Beberapa perintah yang lain juga perlu disertakan di bagian *preamble* untuk keperluan-keperluan khusus tetapi belum dibicarakan untuk saat sini. Perintah-perintah itu akan dibicarakan nanti pada bagian-bagian tertentu sesuai topik bahasan.

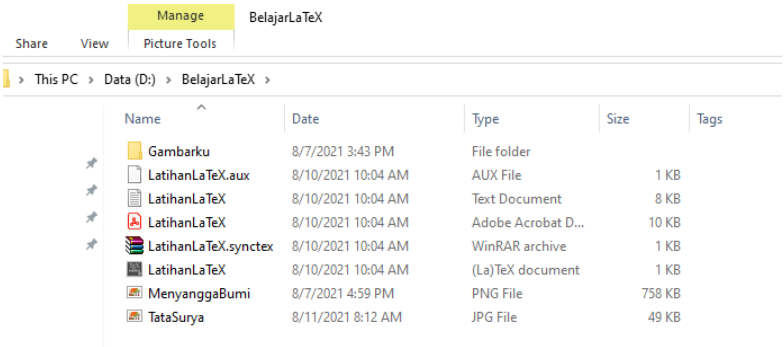
4.1 Cara Sederhana Memasukkan Gambar

Pastikan *package* `graphicx` telah ditambahkan! Kita akan memulai dengan membahas cara paling sederhana untuk memasukkan file gambar ke dalam dokumen. Pertama kita simpan terlebih dahulu

file gambar itu di dalam folder tempat file .tex kita berada. Untuk amannya, nama file dari gambar sebaiknya **tidak mengandung spasi dan tidak mengandung titik**. Pemberian spasi dan/atau titik di nama file pada kasus tertentu menyebabkan perintah tidak dapat dimengerti dan dokumen *error* saat dicompile. Selanjutnya perintah untuk memasukkan gambar adalah

```
\includegraphics{NamaFileGambar}
```

dan untuk lebih lengkapnya dapat ditambahkan *optional argument* (akan dibahas nanti).



Gambar 4.1: Tampilan isi folder yang memuat file-file gambar

Andaikan file .tex kita (misal di sini LatihanLaTeX.tex) berada di dalam folder BelajarLaTeX dan di dalam folder ini pula kita menyimpan file-file gambar, misal dicontohkan di sini TataSurya.jpg dan MenyanggaBumi.png seperti pada gambar 4.1, maka kita dapat memanggil gambar TataSurya.jpg dengan menuliskan perintah `\includegraphics{TataSurya.jpg}` atau tanpa menuliskan ekstensinya, yakni `\includegraphics{TataSurya}` saja (tanpa .jpg).

Apa pengaruh menuliskan .jpg dengan tidak menuliskan .jpg? Jika file gambar dengan nama "TataSurya" hanya ada satu di dalam folder maka tidak ada pengaruh menuliskan .jpg atau tidak menuliskan .jpg. Hal yang berbeda ketika ada dua file dengan na-

ma masing-masing adalah TataSurya.jpg dan TataSurya.png, maka LaTeX dengan sendirinya akan memilih salah satunya dan belum tentu sesuai dengan yang kita harapkan.

Bagaimana pula akibatnya jika kita salah mengetikkan huruf kapital dalam menuliskan nama file? Jika kita memanggil file dengan perintah `\includegraphics{tataSURYA.jpg}` misal, LaTeX dengan sendirinya akan mencari file dengan nama sesuai karakter-karakter huruf yang diketikkan meskipun berbeda dari segi kapital atau bukan kapital. Misalkan di dalam folder ada file TataSurya.jpg dan TATASURYA.jpg, LaTeX akan memilih file yang "dirasa" paling sesuai. Untuk amannya, tuliskan sesuai dengan nama file aslinya dengan memperhatikan penggunaan huruf-huruf kapital yang ada di sana! Sertakan pula ekstensi filenya, baik itu .jpg, .png, atau ekstensi yang lain!

Untuk melihat bagaimana memasukkan gambar ke dalam dokumen, perhatikan contoh berikut!

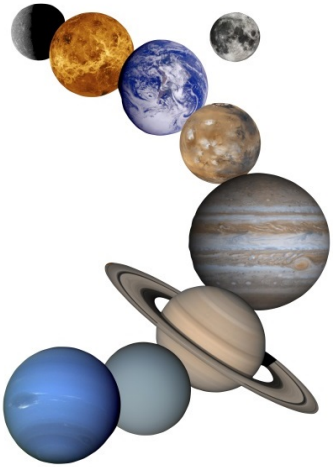
Contoh:
Coba tuliskan kode berikut ini di antara `\begin{document}` dan `\end{document}` (nama file dapat diganti sesuai file gambar yang dipunyai)!

```
Sebuah kenyataan bahwa alam tak mudah dibahasakan.
Ilmu alam, yang fisika adalah bagiannya, "terpaksa"
memilih matematika dengan beragam titik lemah sebagai
bahasanya.  Melalui matematika, fisika memodelkan alam
dengan ungkapan-ungkapan sederhana meskipun banyak
pula yang mengatakan pemodelan itu belum sederhana.
\includegraphics{TataSurya.jpg}
Akan tetapi tak dapat dipungkiri bahwa kebenaran akhir
tak mampu disajikan oleh fisika.  Fisika hanya
mengusulkan sebuah konsep yang untuk sementara diterima
sebagai kebenaran (atau boleh disebut dengan istilah
kebenaran terakhir).  Matematika sebagai bahasa
bertumpu penuh pada logika, sementara logika tak
lebih dari sarana penentu kesahihan, bukan kebenaran
mutlak atau kebenaran sejati.  Di samping itu, dalam
```

```
fungsinya menentukan kesahihan suatu pernyataan,
ambiguitas di dalam logika masih banyak ditemukan.
```

Hasil dari pengetikan kode di atas adalah sebagai berikut:

Sebuah kenyataan bahwa alam tak mudah dibahasakan. Ilmu alam, yang fisika adalah bagiannya, "terpaksa" memilih matematika dengan beragam titik lemah sebagai bahasanya. Melalui matematika, fisika memodelkan alam dengan ungkapan-ungkapan sederhana meskipun banyak pula yang mengatakan pemodelan itu belum sederhana. Akan tetapi tak dapat dipungkiri bahwa kebenaran akhir tak mampu disajikan oleh fisika. Fisika hanya mengusulkan sebuah konsep yang untuk sementara diterima sebagai kebenaran (atau boleh disebut dengan istilah kebenaran terakhir). Matematika sebagai bahasa bertumpu penuh pada logika, sementara logika tak lebih dari sarana penentu kesahihan, bukan kebenaran mutlak atau kebenaran sejati. Di samping itu, dalam fungsinya menentukan kesahihan suatu pernyataan, ambiguitas di dalam logika masih banyak ditemukan.



Perhatikan teks beserta gambar di atas! Luaran yang dihasilkan memperlihatkan bahwa ukuran gambar yang muncul adalah sesuai dengan ukuran gambar aslinya. Letak gambar belum ditentukan secara presisi tetapi biasanya akan disesuaikan dengan tempat kita mengetikkan perintah `\includegraphics` (meskipun tidak selalu, bergantung pada tampilan dokumen yang kita buat) dan biasanya **tidak rapi**.

4.2 *Environment* `figure` dan Letak Gambar

Selanjutnya kita bisa mengatur letak gambar sesuai dengan posisi yang kita inginkan. Caranya adalah dengan menggunakan *environment* `figure`. Perintah untuk memasukkan gambar seperti yang telah dibicarakan sebelumnya akan dituliskan di dalam *envoronment* ini. Letak gambar ditentukan melalui *optional argument*. Lebih jelasnya perhatikan cara penulisan kodenya berikut ini!

```
\begin{figure}[?]
\icludegraphics{NamaFileGambar}
\end{figure}
```

Tanda tanya (`?`) di *optional argument* diganti salah satu dari `t, b, h` atau `h!`. `t` artinya atas (*top*), yakni gambar diletakkan di bagian atas halaman, `b` artinya bawah (*bottom*), `h` dan `h!` artinya gambar diletakkan sesuai tempat penulisan perintah (*here*). Makna `h` dan `h!` hampir sama, perbedaannya adalah `h` masih mengijinkan LaTeX untuk mencarikan posisi terbaik dari gambar itu di sekitar "di sini". Jadi meskipun perintahnya adalah *here*, jika di posisi itu tepat "dirasa" kurang pas bagi LaTeX , LaTeX akan turut membantu menentukan posisinya. Berbeda halnya dengan `h!`, gambar harus berada tepat benar-benar "di sini".

Gambar juga dapat diletakkan di sebelah kiri, tengah, ataupun kanan (dalam arah horisontal) dengan cara mengapit perintah `\icludegraphics{NamaFileGambar}` berturut-turut dengan *environment-environment*: `\begin{flushleft}...\end{flushleft}`,

`\begin{center}...\end{center}`, atau `\begin{flushright}...\end{flushright}`. Cara lain yang lebih sederhana jika ingin meletakkan gambar di tengah adalah menggunakan perintah `\centering` di dalam *environment* `figure`.

Contoh:
Tuliskan kode berikut ini (nama file dapat diganti sesuai file gambar yang dimiliki)!

```
Sebuah kenyataan bahwa alam tak mudah dibahasakan.
Ilmu alam, yang fisika adalah bagiannya, "terpaksa"
memilih matematika dengan beragam titik lemah sebagai
bahasanya.  Melalui matematika, fisika memodelkan alam
dengan ungkapan-ungkapan sederhana meskipun banyak
pula yang mengatakan pemodelan itu belum sederhana.
\begin{figure}[?]
\centering
\icludegraphics{MenyanggaBumi.png}
\end{figure}
Akan tetapi tak dapat dipungkiri bahwa kebenaran akhir
tak mampu disajikan oleh fisika.  Fisika hanya
mengusulkan sebuah konsep yang untuk sementara diterima
sebagai kebenaran (atau boleh disebut dengan istilah
kebenaran terakhir).  Matematika sebagai bahasa
bertumpu penuh pada logika, sementara logika tak
lebih dari sarana penentu kesahihan, bukan kebenaran
mutlak atau kebenaran sejati.  Di samping itu, dalam
fungsinya menentukan kesahihan suatu pernyataan,
ambiguitas di dalam logika masih banyak ditemukan.
```

Kode di atas jika dicompile hasilnya akan *error* karena *optional argument* masih berisi tanda tanya. Kita akan mengganti `?` berturut-turut dengan `t` dan `h!` (`b` dan `h` silakan dipergunakan untuk latihan). Hasilnya akan diperlihatkan di halaman-halaman berbeda setelah halaman ini agar kita dapat melihat betul posisi gambar di halaman tempat teks berada. Silakan perhatikan hasilnya di dua halaman ke belakang!

Sebuah kenyataan bahwa alam tak mudah dibahasakan. Ilmu alam, yang fisika adalah bagiannya, "terpaksa" memilih matematika dengan beragam titik lemah sebagai bahasanya. Melalui matematika, fisika memodelkan alam dengan ungkapan-ungkapan sederhana meskipun banyak pula yang mengatakan pemodelan itu belum sederhana. Akan tetapi tak dapat dipungkiri bahwa kebenaran akhir tak mampu disajikan oleh fisika. Fisika hanya mengusulkan sebuah konsep yang untuk sementara diterima sebagai kebenaran (atau boleh disebut dengan istilah kebenaran terakhir). Matematika sebagai bahasa bertumpu penuh pada logika, sementara logika tak lebih dari sarana penentu kesahihan, bukan kebenaran mutlak atau kebenaran sejati. Di samping itu, dalam fungsinya menentukan kesahihan suatu pernyataan, ambiguitas di dalam logika masih banyak ditemukan.

Sebuah kenyataan bahwa alam tak mudah dibahasakan. Ilmu alam, yang fisika adalah bagiannya, "terpaksa" memilih matematika dengan beragam titik lemah sebagai bahasanya. Melalui matematika, fisika memodelkan alam dengan ungkapan-ungkapan sederhana meskipun banyak pula yang mengatakan pemodelan itu belum sederhana. Akan tetapi tak dapat dipungkiri bahwa kebenaran akhir

tak mampu disajikan oleh fisika. Fisika hanya mengusulkan sebuah konsep yang untuk sementara diterima sebagai kebenaran (atau boleh disebut dengan istilah kebenaran terakhir). Matematika sebagai bahasa bertumpu penuh pada logika, sementara logika tak lebih dari sarana penentu kesahihan, bukan kebenaran mutlak atau kebenaran sejati. Di samping itu, dalam fungsinya menentukan kesahihan suatu pernyataan, ambiguitas di dalam logika masih banyak ditemukan.

4.3 Skala, Ukuran Gambar, dan Rotasi

Perintah untuk memasukkan gambar dapat disertai dengan *optional argument*. Melalui *optional argument*, skala, ukuran gambar, dan sudut rotasi gambar dapat diatur sesuai kehendak kita. Titik-titik di dalam kurung kotak pada perintah

```
\includegraphics[...]{NamaFileGambar}
```

dapat diisi dengan pengaturan skala, ukuran gambar, dan rotasi, yang masing-masing dipisahkan dengan koma (`,`).

Pengaturan skala dilakukan dengan menuliskan `scale=besar skala`. Besar skala diisi dengan bilangan positif tertentu dan bisa desimal. Misal kita menuliskan `scale=0.5`, ini artinya kita akan memasukkan gambar ke dalam dokumen dengan ukuran (lebar dan tinggi) setengah kali ukuran aslinya.

Contoh:
Andaikan kita memiliki gambar Mikroskop.jpg dengan ukuran asli seperti ditampakkan pada gambar 4.2, maka penulisan kode berikut

Gambar 4.2: Gambar Mikroskop.jpg dalam ukuran aslinya

```
\includegraphics[scale=0.75]{Mikroskop.jpg}
```

akan menghasilkan gambar dengan ukuran seperti ditampilkan pada gambar 4.3.

Gambar 4.3: Gambar Mikroskop.jpg dengan skala 0.75

Kita dapat pula mengatur sendiri lebar gambar dengan menggunakan `width=lebar gambar` dan tinggi gambar diatur menggunakan `height=tinggi gambar`. Baik `lebar gambar` ataupun `tinggi gambar` dapat diganti dengan panjang beserta satuan yang kita inginkan, misal `4cm` atau yang lain, dan bisa juga diganti dengan salah satu dari perintah `\textwidth` (lebar teks), `\textheigt` (tinggi teks), `\paperwidth` (lebar kertas), dan `\paperheight` (tinggi kertas). Jika kita menginginkan panjang seperempat kali lebar kertas, dapat juga kita menuliskan `0.25\paperwidth`, demikian juga yang lainnya. Sebagai tambahan, jika lebar dan/atau tinggi gambar telah diatur, pengaturan skala (menggunakan `scale`) menjadi tidak berfungsi.

Contoh:

1. Untuk file gambar Mikroskop.jpg, jika diinputkan perintah

   ```
   \includegraphics[height=0.5\textheight]
   {Mikroskop.jpg}
   ```

maka ukuran gambar yang dihasilkan adalah seperti pada gambar 4.4. Perhatikan bahwa jika tinggi gambar diatur secara manual tetapi lebar gambar tidak diatur maka lebar gambar akan menyesuaikan sesuai perbandingan lebar dan tinggi gambar aslinya.

Gambar 4.4: Gambar Mikroskop.jpg dengan tinggi setengah kali tinggi teks

2. Untuk file gambar Mikroskop.jpg, jika diinputkan perintah

   ```
   \includegraphics[width=4cm,height=6cm]
   {Mikroskop.jpg}
   ```

   maka ukuran gambar yang dihasilkan adalah seperti pada gambar 4.5. Gambar menjadi tidak proporsional karena perbandingan lebar dan tinggi gambar berubah dari ukuran aslinya.

Gambar 4.5: Gambar Mikroskop.jpg dengan lebar 4 cm dan tinggi 6 cm

Berikutnya dengan menambahkan `angle=sudut putar` di *optional argument*, gambar dapat juga dirotasikan sebesar sudut tertentu. `sudut putar` diganti dengan bilangan yang menyatakan besarnya perputaran dalam derajat berlawanan dengan arah putar jarum jam. `sudut putar` boleh negatif, yang berarti arah putarnya berkebalikan, yakni searah dengan arah putar jarum jam.

Contoh:
Dengan kode berikut ini gambar Mikroskop.jpg akan diputar -30 derajat (30 derajat searah dengan arah putar jarum jam) sebagaimana gambar 4.6:

```
\includegraphics[angle=-30]{Mikroskop.jpg}
```

4.4 Menambahkan *Caption*

Beberapa gambar di depan telah ditampilkan dengan *caption* yang menyertainya tetapi secara khusus kita belum membahas tentang bagaimana cara menambahkan *caption*. Cara menambahkan *cap-*

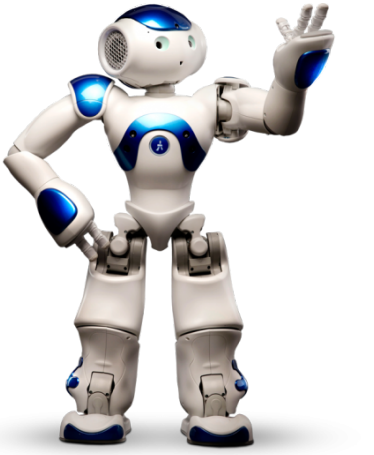
Gambar 4.6: Gambar Mikroskop.jpg yang diputar −30 derajat

tion adalah dengan menuliskan perintah `\caption{...}` (ganti titik-titik dengan *caption* yang ingin dituliskan) di dalam *environment* `figure`. Dengan kata lain, untuk menambahkan *caption* kita harus terlebih dulu menggunakan *environment* `figure`. *Caption* akan ditampilkan lengkap dengan nomor gambar secara otomatis.

Contoh:
Gambar RobotPintar.png akan dimasukkan ke dalam dokumen dengan *caption* "Robot pintar era industri 4.0" seperti pada gambar 4.7. Kode yang perlu dituliskan adalah

```
\begin{figure}
\centering
\includegraphics{RobotPintar.png}
```

```
\caption{Robot pintar era industri 4.0}
\end{figure}
```



Gambar 4.7: Robot pintar era industri 4.0

4.5 Menambahkan Label Gambar

Perlu diperhatikan bahwa (meskipun disebut) label bagi gambar, label itu sesungguhnya adalah label bagi *caption*. Oleh karenanya untuk memberikan label bagi gambar, tuliskan perintahnya setelah menuliskan *caption*! Perintah untuk menulis label selalu sama, yaitu `\label{Nama label}`. Untuk memanggilnya dari tempat lain, digunakan perintah `\ref{Nama label}` (hasilnya akan muncul setelah ***dicompile*** **dua kali**).

Contoh:
Perhatikan teks berikut gambar yang menyertainya di bawah ini!

Stopwatch terbagi atas jenis analog dan digital. Gambar 4.8 mem-

perlihatkan sebuah contoh stopwatch analog.

Gambar 4.8: Stopwatch

Kode yang perlu dituliskan untuk menghasilkan luaran seperti di atas adalah

```
Stopwatch terbagi atas jenis analog dan digital.
Gambar \ref{stpwtch} memperlihatkan sebuah contoh
stopwatch analog.
\begin{figure}[h!]
\centering
\includegraphics{Stopwatch.png}
\caption{Stopwatch}
\label{stpwtch}
end{figure}
```

4.6 Memasukkan Gambar dari Suatu Folder

Kita telah belajar cara memasukkan file gambar yang berada di dalam folder tempat file .tex kita berada ke dalam dokumen. Cara ini memang mudah tetapi jika banyak file gambar yang kita gunakan

dan seluruh file gambar berada di folder itu, tentu isi folder menjadi "berantakan" dengan beragam jenis file ada di sana tanpa penataan.

Untuk memasukkan file-file gambar yang berada di folder lain ke dalam dokumen, perlu ditambahkan sebuah perintah pada *preamble* dan **dituliskan setelah** `\usepackage{graphicx}` agar LaTeX mengerti alamat folder (*path*) tempat gambar yang nanti kita panggil. Perintahnya adalah

```
\graphicspath{{...}}
```

Titik-titik diisi dengan alamat folder (yang berisi file-file gambar yang hendak kita panggil). Dengan memasukkan perintah di atas, setiap kali kita memanggil gambar, LaTeX **hanya** akan mencari file gambar di alamat folder yang telah dituliskan dan di folder tempat file .tex berada.

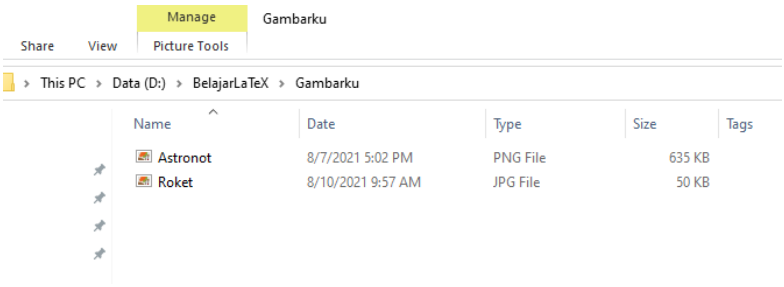
Tidak harus dalam satu folder, file-file gambar kita juga boleh berada di beberapa folder berbeda. Jika ada sejumlah n folder maka n buah *path* harus dituliskan, masing-masing berada di dalam kurung kurawal yang berbeda. Sebagai contoh jika ada tiga folder, perintah di *preamble* menjadi

```
\graphicspath{{...}{...}{...}}
```

(Tiga titik-titik diisi dengan tiga *path*). Ada aturan khusus tentang cara menulis *path* bergantung pada letak atau alamat folder. Hal yang penting untuk dicatat bahwa semua nama folder yang nanti kita tuliskan saat membuat kode **tidak boleh mengandung spasi ataupun titik**.

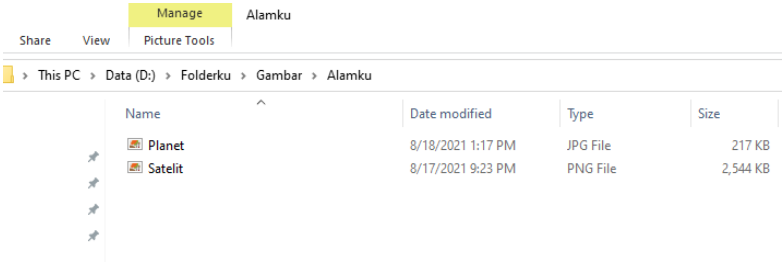
Kita akan mulai dengan cara menulis *path* untuk folder berisi file-file gambar yang ada di dalam folder tempat file .tex kita berada. Kembali perhatikan gambar 4.1! File LatihanLaTeX.tex berada di dalam folder BelajarLaTeX. Folder Gambarku juga berada di dalam folder BelajarLaTeX. Folder Gambarku berisi dua file gambar, yakni Astronot.png dan Roket.jpg, seperti diperlihatkan pada gambar 4.9. *Path* folder Gambarku dapat ditulis secara singkat dengan kode `./Gambarku/` (perhatikan ada titik sebelum garis miring pertama dan ingat bahwa garis miring kedua harus dituliskan) sehingga di *preamble* kita menuliskan `\graphicspath{{./Gambarku/}}`.

Penggunaan titik (.) sudah mewakili *path* folder BelajarLatex (cara singkat).



Gambar 4.9: Isi folder Gambarku

Jika file-file gambar ada di folder lain atau tidak di dalam folder tempat file .tex kita berada maka *path* folder harus dituliskan secara lengkap. Sebagai contoh, file-file gambar (Planet.jpg dan Satelit.png) ada di suatu folder dengan nama folder Alamku, seperti pada gambar 4.10. Dalam hal ini perintah di *preamble* yang harus ditulis adalah `\graphicspath{{D:/Folderku/Gambar/Alamku/}}`. Jangan lupa menulis garis miring terakhir di *path*!



Gambar 4.10: Isi folder Alamku

Apabila file-file gambar yang akan kita panggil untuk dimasukkan ke dokumen berada di dua folder, yakni Gambarku dan Alamku (dengan dua *path* yang telah disebutkan), maka di *preamble* harus dituliskan

```
\graphicspath{{./Gambarku/}{D:/Folderku/Gambar/Alamku/}}
```

Selanjutnya perintah untuk memasukkan gambar adalah sama seperti yang telah dibahas di muka, yaitu menggunakan perintah `\includegraphics{NamaFileGambar}` dan dapat disertai dengan *environment* `figure`. Perihal ini tidak perlu diulang pembahasannya di sini, silakan dipergunakan sebagai bahan latihan mandiri!

Bab 5

Menambahkan Tabel

Untuk pembuatan tabel yang (sangat) sederhana, LaTeX tidak membutuhkan tambahan *package* tertentu. Akan tetapi untuk keperluan penggabungan baris dan berbagai modifikasi tabel, beberapa *package* tertentu mutlak harus disertakan. Beberapa *package* yang sekiranya dibutuhkan dalam pembuatan tabel berikut perintah-perintah bawaannya akan dijelaskan di dalam subbab-subbab sesuai topik bahasan.

5.1 Membuat Tabel Sederhana

Untuk membuat tabel digunakan *environment* `tabular`. Padanya terdapat *mandatory argument* yang harus diisikan untuk menentukan jumlah kolom berikut aturan perataan teks di setiap kolom. Lebih jelasnya adalah sebagai berikut:

```
\begin{tabular}{...}
Isi tabel
\end{tabular}
```

Titik-titik di dalam kurung kurawal (*mandatory argument*) diisi dengan sederetan huruf yang masing-masing hurufnya adalah salah satu dari `l, c,` dan `r` (ada opsi huruf-huruf lain tetapi belum dije-

laskan pada bagian ini). Huruf `l` kependekan dari *left*, yakni untuk mengatur teks di dalam kolom menjadi rata kiri. Huruf `c` kependekan dari *center*, untuk mengatur teks di dalam kolom menjadi rata tengah. Huruf `r` kependekan dari *right*, untuk mengatur teks di dalam kolom menjadi rata kanan. Misalkan kita menulis {`cll`}, ini artinya kita hendak membuat tabel 3 kolom dengan kolom pertama, kedua, dan ketiga berturut-turut isinya (teks di dalamnya) dibuat rata tengah, rata kiri, dan rata kiri.

Membuat garis pada tabel ada aturannya tersendiri. Untuk menambahkan garis vertikal, pada *mandatory argument* perlu disertakan `|` dan ditempatkan sesuai bagian tabel atau urutan kolom yang di sana kita ingin membuat garis. Misal kita menulis {`|c|ll`}, ini artinya garis vertikal dibuat hanya untuk mengapit kolom pertama. Garis vertikal dapat dibuat rangkap dua dengan menuliskan `||`. Untuk garis horisontal, perintah untuk membuatnya adalah pada bagian `isi tabel` (cara menuliskan isi tabel akan dibahas lebih dulu).

Pada bagian `isi tabel`, untuk berpindah kolom digunakan kode `&` dan untuk berpindah baris digunakan perintah \\ (bisa disertai spasi vertikal, misal \\`[0.5cm]`). Selanjutnya jika kita ingin memindahkan baris teks pada bagian tertentu tetapi masih di baris tabel atau sel yang sama, perintah \\ tidak lagi bisa digunakan. Bukan menyebabkan *error*, tetapi perintah \\ telah memiliki kegunaan khusus, yakni untuk berpindah baris pada tabel, bukan memindahkan baris teks. Untuk memindahkan baris teks, digunakan perintah `\newline`.

Garis horisontal dapat dibuat dengan perintah `\hline` dan tempat penulisannya adalah sesuai urutan baris yang kita inginkan. Misal sebelum menuliskan isi baris pertama dituliskan `\hline` terlebih dulu, ini berarti kita sedang memasukkan perintah untuk membuat garis horisontal di bagian paling atas dari tabel. Misalkan lagi kita menulis perintah `\hline` setelah menuliskan \\ yang pertama, ini artinya kita sedang membuat garis horisontal yang memisahkan baris pertama dan kedua. Jika perintah `\hline` ditulis dua kali (berurutan) maka garis horisontal yang terbentuk menjadi rangkap dua.

Contoh:
Mari kita membuat tabel sederhana! Perhatikan tabel yang menyebutkan nama beberapa besaran fisika dan satuan-satuannya di bawah ini!

No.	Nama Besaran	Satuan
1	Jarak	meter
2	Waktu	detik
3	Kelajuan	meter/detik

Tabel di atas dibuat dengan menuliskan kode:

```
\begin{tabular}{c|l|l}
\hline
No.&Nama Besaran&Satuan\\
\hline
1&Jarak&meter\\
2&Waktu&detik\\
3&Kelajuan&meter/detik\\
\hline
\end{tabular}
```

Selain perintah `\hline` ada pula perintah `\cline{m-n}`. Perintah ini digunakan untuk membuat garis horisontal dari kolom ke-`m` hingga kolom ke-`n`, jadi tidak harus mengenai semua kolom. Ganti `m` dan `n` dengan bilangan-bilangan yang menyatakan nomor-nomor kolom!

Contoh:
Perhatikan tabel di bawah ini! Pada tabel di bawah ini, garis-garis horisontal di bawah baris kedua dan ketiga tidak mengenai kolom paling kiri.

No.	Nama Besaran	Satuan
1	Jarak	meter
2	Waktu	detik
3	Kelajuan	meter/detik

Tabel di atas dibuat dengan menuliskan kode:

```
\begin{tabular}{c|l|l}
\hline
No.&Nama Besaran&Satuan\\
\hline
1&Jarak&meter\\
\cline{2-3}
2&Waktu&detik\\
\cline{2-3}
3&Kelajuan&meter/detik\\
\hline
\end{tabular}
```

5.2 Letak Tabel dan *Caption*

Tabel sederhana yang telah kita buat di subbab 5.1 belum diatur secara khusus penempatannya. Untuk mengatur letak tabel, diperlukan *environment* yang memiliki kegunaan hampir serupa dengan *environment* `figure` (yang bisa digunakan untuk mengatur letak sekaligus memberi *caption* dan label pada gambar). *Environment* itu adalah `table` (penulisan secara lengkapnya yaitu `\begin{table}...\end{table}`). *Environment* `tabular` dituliskan di dalam *environment* `table`. *Optional argument* yang sama, yakni `t, b, h, dan h!`, dapat digunakan untuk mengatur letak tabel (dalam arah vertikal) dalam dokumen. Untuk mengatur letak tabel dalam arah horisontal, perintahnya juga sama seperti pada penjelasan tentang *environment* `figure` di subbab 4.2 (pembaca diasumsikan telah mahir). Misal, kita ingin menempatkan tabel di tengah dalam arah horisontal, dapat digunakan perintah `\centering`.

Untuk menambahkan *caption*, perintah yang digunakan juga sama, yakni `\caption{...}`. Perhatikan urutan penulisan! Jika perintah `\caption` dituliskan sebelum *environment* `tabular`, *caption* akan ditempatkan sebelum (di atas) tabel. Sebaliknya jika perintah `\caption` dituliskan setelah *environment* `tabular`, *caption* akan ditempatkan setelah (di bawah) tabel. Sebagai tambahan, perintah untuk membuat label juga dapat disertakan (dituliskan) setelah perintah `\caption`.

Contoh:
Perhatikan tabel 5.1! Tabel ini telah diatur penempatannya serta

Tabel 5.1: Beberapa besaran fisika dan satuan-satuannya

No.	Nama Besaran	Satuan
1	Jarak	meter
2	Waktu	detik
3	Kelajuan	meter/detik

telah diberi *caption* di atas tabel. Kode yang dituliskan adalah

```
\begin{table}[h!]
\centering
\caption{Beberapa besaran fisika dan satuan-satuannya}
\vspace{0.3cm}
\begin{tabular}{c|l|l}
\hline
No.&Nama Besaran&Satuan\\
\hline
1&Jarak&meter\\
2&Waktu&detik\\
3&Kelajuan&meter/detik\\
\hline
\end{tabular}
\end{table}
```

Perhatikan bahwa di dalam kode yang dituliskan terdapat perintah `\vspace{0.3cm}`. Ini tidak wajib tetapi dapat kita tuliskan jika kita ingin memberikan spasi vertikal antara *caption* (yang dalam hal ini diletakkan di atas) dengan tabel. Tanpa memberikan spasi vertikal biasanya jaraknya terlalu kecil atau terlalu dekat.

5.3 *Text Wrapping* pada Tabel

Secara *default*, teks-teks yang ditulis di dalam tabel tidak dibatasi lebarnya dan lebar kolom akan disesuaikan dengan panjang teks

yang ada di dalamnya. Akibatnya, jika teks terlalu panjang maka lebar tabel bisa melebihi lebar halaman. Masalah ini dapat diatasi dengan membatasi lebar teks. Jika lebar teks sudah diatur, teks yang terlalu panjang otomatis akan dibuat menjadi beberapa baris.

Mandatory argument pada *environment* `tabular` selain berisi deretan huruf `l, c,` dan `r`, untuk keperluan pembatasan lebar teks juga bisa berisi opsi `p{lebar teks}, m{lebar teks}, dan b{lebar teks}`. Untuk dua opsi terakhir, yakni `m` dan `b`, keduanya dapat dituliskan jika terlebih dulu *package* `array` telah ditambahkan di *preamble* (tuliskan di *preamble* `\usepackage{array}`!). Kode `p` digunakan jika teks ingin dibuat rata di bagian atas (relatif terhadap teks-teks di kolom-kolom lain), kode `m` digunakan jika teks ingin dibuat rata di bagian tengah (secara vertikal), dan kode `b` digunakan jika teks ingin dibuat rata di bagian bawah. Ganti pula `lebar teks` dengan lebar teks yang diinginkan (beserta satuannya) pada kolom yang kita buat!

Contoh:
Perhatikan tabel 5.2! Tabel ini adalah hasil atau luaran dari penu-

Tabel 5.2: Pengertian bilangan rasional dan irasional

No.	Istilah	Pengertian
1	Bilangan rasional	Bilangan riil yang dapat dinyatakan dalam $\frac{a}{b}$, dengan a dan b bilangan-bilangan bulat.
2	Bilangan irasional	Bilangan riil yang tidak dapat dinyatakan dalam $\frac{a}{b}$, dengan a dan b bilangan-bilangan bulat.

lisan kode berikut:

```
\begin{table}[h!]
\centering
\caption{Pengertian bilangan rasional dan irasional}
```

```
\vspace{0.3cm}
\begin{tabular}{|c|l|p{4cm}|}
\hline\hline
No.&Istilah&Pengertian\\
\hline\hline
1&Bilangan rasional&Bilangan riil yang dapat dinyatakan
dalam $\frac{a}{b}$, dengan $a$ dan $b$ bilangan-
bilangan bulat.\\
\hline
2&Bilangan irasional&Bilangan riil yang tidak dapat
dinyatakan dalam $\frac{a}{b}$, dengan $a$ dan $b$
bilangan-bilangan bulat.\\
\hline\hline
\end{tabular}
\end{table}
```

Jika pembatasan lebar teks sekaligus ingin disertai pengaturan rata kiri, tengah, atau rata kanan, maka sebelum menuliskan `p, m,` dan `b` tuliskan terlebih dahulu `>{\raggedright\arraybackslash}` (untuk rata kiri), `>{\centering\arraybackslash}` (untuk rata tengah), atau `>{\raggedleft\arraybackslash}` (untuk rata kanan)! Misalkan kita ingin membuat suatu kolom dengan lebar teks 4 cm menggunakan `p` dan rata tengah secara horisontal, yang harus kita tuliskan di *mandatory argument* pada *environment* `tabular` adalah `>{\centering\arraybackslash}p{4cm}`. Perlu diperhatikan bahwa untuk kebutuhan ini, *package* `array` wajib sudah ditambahkan di *preamble*.

5.4 Penggabungan Baris dan Kolom

Beberapa baris dan/atau kolom dapat digabungkan (*merged*). Penggabungan kolom (saja) tidak memerlukan tambahan *package* tertentu, sementara penggabungan baris membutuhkan tambahan *package* `multirow`. "Prinsip" penggabungan kolom berbeda dengan penggabungan baris. Bahasan tentang ini akan diberikan satu per satu.

"Prinsip" penggabungan kolom adalah beberapa kolom pada su-

atu baris tertentu benar-benar dibuat menjadi satu kolom baru. Artinya, jika di awal kita bisa memasukkan teks-teks ke dua buah kolom (di suatu baris), setelah keduanya digabung kita hanya bisa menuliskan teks ke satu buah kolom. Perintah untuk menggabungkan `n` buah kolom adalah

```
\multicolumn{n}{...}{teks}
```

Perintah ini dituliskan di tempat menuliskan isi `n` kolom yang digabung. Titik-titik diisi `l, c, r, p{lebar teks}, m{lebar teks},` atau `b{lebar teks}` untuk keperluan perataan teks di dalam sel hasil gabungan. Kita dapat pula menyertakan perintah untuk membuat garis vertikal `|`. Khusus untuk *text wrapping*, kita juga dapat menyertakan pengaturan rata kiri, tengah, atau kanan, sebagaimana telah dibicarakan di subbab 5.3.

Contoh:
Luaran dari kode berikut ini adalah tabel 5.3:

```
\begin{table}[h!]
\centering
\caption{Daftar nilai tugas mahasiswa}
\vspace{0.3cm}
\begin{tabular}{|c|l|c|}
\hline
No.&Nama mahasiswa&Nilai\\
\hline
1&Monica&80\\
2&Yogi&85\\
3&Umam&90\\
\hline
\multicolumn{2}{|r|}{Nilai rerata}&85\\
\hline
\end{tabular}
\end{table}
```

Cermati bahwa kode yang digunakan untuk membuat baris terakhir tabel hanya membutuhkan satu buah `&`, berbeda dengan baris lain yang membutuhkan dua buah `&`! Ini karena dua buah kolom di baris

Tabel 5.3: Daftar nilai tugas mahasiswa

No.	Nama mahasiswa	Nilai
1	Monica	80
2	Yogi	85
3	Umam	90
	Nilai rerata	85

terakhir telah diubah menjadi satu buah kolom.

Selanjutnya untuk keperluan penggabungan baris, jangan lupa di *preamble* tuliskan terlebih dulu `\usepackage{multirow}`! "Prinsip" penggabungan `n` buah baris pada kolom tertentu adalah `n` buah baris itu tetap **dipertahankan keberadaannya** tetapi baris paling atas dari baris-baris yang digabungkan itu dibuat memanjang ke bawah (menjadi tinggi) sehingga menutup baris-baris lain yang ada di bawahnya. Karena dalam hal ini baris-baris yang lain itu tetap ada, kita masih bisa memasukkan teks-teks ke masing-masing baris itu dan berakibat teks-teks menjadi bertabrakan. Dari sini muncul semacam "kewajiban" untuk mengosongkan isi baris-baris yang lain. Demikian juga dengan perintah `\hline`, garis lurus mendatar yang dihasilkan bisa menabrak sel hasil gabungan baris (termasuk menabrak teks yang ada di dalamnya). Oleh karena itu perintah `\cdot` menjadi diperlukan.

Perintah untuk menggabungkan `n` buah baris adalah sebagai berikut:

```
\multirow{n}{lebar teks}{teks}
```

Ganti `lebar teks` sesuai dengan lebar teks (beserta satuannya) di sel hasil gabungan yang diinginkan! Jika tidak menginginkan ada pengaturan terkait lebar teks maka ganti `lebar teks` dengan `*`. Perintah penggabungan baris ditulis di bagian teks pada baris paling atas di antara baris-baris yang digabung.

Contoh:
Tabel 5.4 dibuat menggunakan kode berikut:

```
\begin{table}[h!]
\centering
\caption{Daftar dosen dan mahasiswa bimbingan}
\vspace{0.3cm}
\begin{tabular}{||m{3.5cm}|m{4.5cm}||}
\hline
\textbf{Nama dosen}&\textbf{Mahasiswa bimbingan}\\
\hline
\multirow{2}{*}{Hartono, M.Sc.}&Prima Vitra Varecha\\
\cline{2-2}
&Muhammad Harun\\
\hline
\multirow{3}{*}{Heni Sumarti, M.Si.}&Nur Chikmi Azizi\\
\cline{2-2}
&Siska Nuryani\\
\cline{2-2}
&Tria Nurmar'atin\\
\hline
\end{tabular}
\end{table}
```

Tabel 5.4: Daftar dosen dan mahasiswa bimbingan

Nama dosen	**Mahasiswa bimbingan**
Hartono, M.Sc.	Prima Vitra Varecha
	Muhammad Harun
Heni Sumarti, M.Si.	Nur Chikmi Azizi
	Siska Nuryani
	Tria Nurmar'atin

Selanjutnya kita akan melihat apa yang terjadi jika salah satu perintah `\cline{2-2}` diganti dengan `\hline` dan salah satu baris yang harusnya dikosongkan kita isi dengan teks "COBA-COBA". Tuliskan kode berikut!

```
\begin{table}[h!]
\centering
```

```
\caption{Tabel kacau}
\vspace{0.3cm}
\begin{tabular}{||m{3.5cm}|m{4.5cm}||}
\hline
\textbf{Nama dosen}&\textbf{Mahasiswa bimbingan}\\
\hline
\multirow{2}{*}{Hartono, M.Sc.}&Prima Vitra Varecha\\
\hline
COBA-COBA&Muhammad Harun\\
\hline
\multirow{3}{*}{Heni Sumarti, M.Si.}&Nur Chikmi Azizi\\
\cline{2-2}
&Siska Nuryani\\
\cline{2-2}
&Tria Nurmar'atin\\
\hline
\end{tabular}
\end{table}
```

Hasilnya kacau (tabel 5.5).

Tabel 5.5: Tabel kacau

Nama dosen	**Mahasiswa bimbingan**
Hartono, M.Sc.	Prima Vitra Varecha
COBA-COBA	Muhammad Harun
Heni Sumarti, M.Si.	Nur Chikmi Azizi
	Siska Nuryani
	Tria Nurmar'atin

Kita juga bisa menggabungkan kolom dan baris sekaligus. Untuk keperluan ini, perintah penggabungan baris dituliskan pada bagian `teks` di perintah penggabungan kolom (**jangan terbalik!**). Kesatuan perintah itu dituliskan di bagian kolom paling kiri dan baris paling atas dari kolom-kolom dan baris-baris yang digabungkan. Misal kita ingin menggabungkan `n` kolom dan `m` baris menjadi satu buah sel, kode yang harus dituliskan untuk mengisi sel itu adalah

```
\multicolumn{n}{...}{\multirow{m}{lebar teks}{teks}}
```

Contoh:
Tabel 5.6 dibuat menggunakan kode berikut:

```
\begin{table}[h!]
\centering
\caption{Notasi atom}
\vspace{0.3cm}
\begin{tabular}{ll|ll}
\hline
\multicolumn{4}{c}{\textbf{Notasi atom unsur $X$}}\\
\hline\hline
\multicolumn{2}{c|}{\multirow{2}{*}{\LARGE$^A_ZX$}}
&$\Sigma$ proton&= $Z$\\
&&$\Sigma$ elektron&= $Z$\\
\cline{1-2}
$A$ = nomor massa&$Z$ = nomor atom&$\Sigma$ neutron
&= $A-Z$\\
\hline
\end{tabular}
\end{table}
```

Tabel 5.6: Notasi atom

Notasi atom unsur X			
$^A_Z X$		Σ proton	$= Z$
		Σ elektron	$= Z$
A = nomor massa	Z = nomor atom	Σ neutron	$= A - Z$

5.5 Perataan Teks Satu Sel

Perataan teks (rata kiri, tengah, dan kanan) menggunakan `l, c,` dan `r` yang telah dibahas di subbab 5.1 berlaku untuk seluruh baris di tiap-tiap kolom. Bagaimana membuat perataan itu agar berlaku hanya di satu sel?

Ada cara "kreatif" yang dapat kita gunakan untuk mengatur perataan teks dalam satu sel. Bermodalkan pengetahuan tentang penggabungan kolom, yang di dalamnya memuat pengaturan perataan teks pada kolom hasil gabungan, kita bisa "seolah-olah" menggabungkan satu kolom pada baris tertentu menjadi satu kolom lagi (sama saja dengan tidak menggabungkan) dan memanfaatkan bagian pengaturan perataan teksnya. Dengan ini perataan teks akhirnya hanya berlaku untuk satu sel. Jadi, di bagian sel yang ingin diatur perataan teksnya kita hanya cukup menuliskan

`\multicolumn{1}{...}{teks}`

dengan titik-titik diisi `l`, `c`, atau `r`. Bisa juga kita mengisikan `p{lebar teks}`, `m{lebar teks}`, atau `b{lebar teks}` untuk *text wrapping*.

Bahasan ini (menurut penulis) sudah cukup jelas dan oleh karenanya tidak diperlukan contoh untuk bagian ini. Silakan mencoba dan berlatih secara mandiri!

5.6 Mengubah Tinggi Baris

Pastikan terlebih dahulu *package* `array` telah ditambahkan di *preamble*! Selanjutnya untuk mengubah tinggi baris secara serentak (untuk seluruh baris yang ada di tabel) menjadi `n` kali tinggi mula-mula, digunakan perintah

`\renewcommand{\arraystretch}{n}`

Perintah ini dituliskan di dalam *environment* `table` sebelum *environment* `tabular`. Jika dituliskan di luar *environment* `table`, perintah ini akan mengenai setiap tabel (di dalam dokumen) yang kita buat setelahnya.

Contoh:
Tinggi baris pada tabel 5.7 dibuat menjadi tiga kali tinggi baris pada tabel standar. Tabel ini dihasilkan dari penulisan kode sebagai berikut:

```
\begin{table}[h!]
\centering
\caption{Para ilmuwan}
\vspace{0.3cm}
\renewcommand{\arraystretch}{3}
\begin{tabular}{|c|l|l|}
\hline
No.&Nama Ilmuwan&Asal Negara\\
\hline\hline
1&Isaac Newton&Inggris\\
\hline
2&Albert Einstein&Jerman\\
\hline
3&Stephen Hawking&Inggris\\
\hline
\end{tabular}
\end{table}
```

Tabel 5.7: Para ilmuwan

No.	Nama Ilmuwan	Asal Negara
1	Isaac Newton	Inggris
2	Albert Einstein	Jerman
3	Stephen Hawking	Inggris

Untuk mengubah tinggi sebuah baris dalam tabel, orang menggunakan beragam cara. Penulis di sini akan mencoba menunjukkan satu cara yang relatif mudah tetapi dengan bantuan satu bu-

ah *package* lagi, yaitu `makecell`. Tuliskan lebih dulu di *preamble* `\usepackage{makecell}`! Perintah untuk mengubah tinggi sebuah baris (ditulis di salah satu sel pada baris itu) adalah

```
\Gape[spasi atas][spasi bawah]{teks}
```

Ganti `spasi atas` dan `spasi bawah` dengan lebar spasi vertikal di atas dan di bawah teks (sehingga ketinggian baris berubah) beserta satuannya! Kita juga bisa menggunakan hanya satu buah *optional argument* atau satu buah kurung kotak. Dengan ini lebar spasi vertikal di atas dan di bawah teks akan dibuat sama.

Contoh:
Lihat tabel 5.8! Ketinggian baris paling atas dibuat lebih dari baris-

Tabel 5.8: Besaran dan jenisnya

Besaran	Jenis Besaran
Torka	Vektor
Energi kinetik	Skalar
Kecepatan sudut	Vektor

baris yang lain. Kode yang dituliskan adalah

```
\begin{table}[h!]
\centering
\caption{Besaran dan jenisnya}
\vspace{0.3cm}
\begin{tabular}{|c||c|}
\hline
\Gape[0.3cm][0.6cm]{Besaran}&Jenis Besaran\\
\hline
Torka&Vektor\\
\hline
Energi kinetik&Skalar\\
\hline
```

```
Kecepatan sudut&Vektor\\
\hline
\end{tabular}
\end{table}
```

5.7 Tabel Profesional

Banyak buku dan artikel-artikel di berbagai jurnal ilmiah yang memiliki *style* khusus dalam pembuatan tabel. Salah satu yang paling khas adalah tidak adanya penggunaan garis vertikal. *Package* `booktabs` memfasilitasi perintah-perintah untuk membuat tabel semacam ini dengan ketebalan garis-garis horisontal khusus yang berbeda antara garis atas tabel, garis tengah, garis bawah, dan garis pendek yang menghubungkan beberapa kolom.

Tuliskan `\usepackage{booktabs}` di *preamble*! Selanjutnya dalam pembuatan tabel, perintah-perintah khusus untuk membuat garis-garis horisontal meliputi:

1. Garis atas tabel dibuat dengan perintah `\toprule`
2. Garis tengah tabel dibuat dengan perintah `\midrule`
3. Garis bawah tabel dibuat dengan perintah `\bottomrule`
4. Garis pendek dari kolom ke-`m` hingga kolom ke-`n` dibuat dengan perintah `\cmidrule{m-n}`. Jika terdapat dua garis pendek yang bersebelahan, ujung dua garis itu akan bertemu sehingga keduanya terhubung. Agar ujung keduanya tidak menyatu, perintah yang digunakan untuk membuat garis pendek adalah `\cmidrule(lr){m-n}`.

Contoh:
Tabel 5.9 berisikan data koordinat dan massa benda-benda titik serta koordinat pusat massa dan massa totalnya. Tabel ini dibuat menggunakan kode berikut:

```
\begin{table}[h!]
\centering
```

```
\caption{Massa dan koordinat benda-benda titik}
\vspace{0.3cm}
\begin{tabular}{clccc}
\toprule
\multirow{2}{*}{No.}&\multirow{2}{*}{Nama Benda}
&\multicolumn{2}{c}{Koordinat}
&\multirow{2}{*}{Massa (kg)}\\
\cmidrule{3-4}
&&$x$ (cm)&$y$ (cm)&\\
\midrule
1&Benda I&0&160&3\\
2&Benda II&160&0&3\\
3&Benda III&0&0&6\\
\midrule
\multicolumn{2}{l}{Benda gabungan}&$x_{pm}$ = 40 cm
&$y_{pm}$ = 40 cm&$M$ = 12 kg\\
\bottomrule
\end{tabular}
\end{table}
```

Tabel 5.9: Massa dan koordinat benda-benda titik

No.	Nama Benda	Koordinat		Massa (kg)
		x (cm)	y (cm)	
1	Benda I	0	160	3
2	Benda II	160	0	3
3	Benda III	0	0	6
Benda gabungan		$x_{pm} = 40$ cm	$y_{pm} = 40$ cm	$M = 12$ kg

5.8 Modifikasi Garis pada Tabel

Modifikasi yang akan kita bahas meliputi pengaturan ketebalan garis dan pembuatan garis putus-putus.

1. **Pengaturan ketebalan garis**. Banyak cara yang mung-

kin ditempuh untuk mengubah ketebalan garis vertikal maupun horisontal pada tabel, yakni dengan jalan memanfaatkan berbagai *package* tertentu. Penulis lebih tertarik menggunakan *package* `boldline`. Dengan menyertakan perintah `\usepackage{boldline}` di *preamble*, pembuatan garis vertikal (yang biasa menggunakan `|`) agar menjadi `k` kali lebih tebal dapat diganti dengan `V{k}`. Pembuatan garis horisontal (yang biasa menggunakan `\hline` dan `\cline{m-n}`) agar menjadi `k` kali lebih tebal dapat diganti berturut-turut dengan menggunakan `\hlineB{k}` dan `\clineB{m-n}{k}`. Ganti `k` dengan bilangan!

2. **Pembuatan garis putus-putus**. Untuk keperluan ini, kita bisa menggunakan *package* `arydshln`. Tuliskan terlebih dulu di *preamble* `\usepackage{arydshln}`! Berikutnya untuk membuat garis vertikal putus-putus kodenya adalah titik dua (`:`) dan untuk membuat garis horisontal putus-putus kodenya adalah `\hdashline` dan `\cdashline{m-n}` (jika garis hanya dari kolom `m` hingga kolom `n`). Panjang strip dan jarak antar strip pada garis putus-putus mendatar dapat diatur dengan cara menambahkan *optional argument* `[panjang strip/jarak antar strip]` tepat setelah perintah membuat garis. Ganti `panjang strip` dan `jarak antar strip` dengan bilangan lengkap dengan satuan, misal `[4pt/2pt]`!

Contoh:
Pastikan *package* `boldline` dan `arydshln` telah ditambahkan di *preamble*! Tabel 5.10 selanjutnya dapat dibuat menggunakan kode berikut:

```
\begin{table}[h!]
\centering
\caption{Tetapan-tetapan fisika}
\vspace{0.3cm}
\renewcommand{\arraystretch}{1.5}
\begin{tabular}{lV{3}l:cV{3}l}
\hlineB{3}
No.&\multicolumn{2}{lV{3}}{Nama Tetapan}&Nilai Tetapan\\
```

```
\hlineB{3}
1&Kelajuan cahaya&$(c)$&$3\times 10^8$ m/s\\
\cdashline{2-4}
2&Bilangan Avogadro&$(N_A)$&$6,02\times 10^{23}$
partikel/mol\\
\cdashline{2-4}
3&Tetapan Planck&$(h)$&$6,63\times 10^{-34}$ Js\\
\hlineB{3}
\end{tabular}
\end{table}
```

Tabel 5.10: Tetapan-tetapan fisika

No.	Nama Tetapan		Nilai Tetapan
1	Kelajuan cahaya	(c)	3×10^8 m/s
2	Bilangan Avogadro	(N_A)	$6,02 \times 10^{23}$ partikel/mol
3	Tetapan Planck	(h)	$6,63 \times 10^{-34}$ Js

5.9 *Environment* `array`

Subbab ini membahas sebuah *environment* dalam teks matematis yang mirip dengan *environment* `tabular` dan berfungsi untuk membuat tabel atau meratakan bagian-bagian tertentu pada persamaan-persamaan matematis. *Environment* `array` dapat dibuat sebagai bagian dari teks matematis dan harus berada di salah satu dari mode *inline* atau *displaymath*. Pengaturan-pengaturan standar yang berlaku di *environment* `tabular` berlaku juga di *environment* `array`.

Contoh:
Perhatikan contoh penurunan persamaan dan pertidaksamaan be-

rikut!

$$\begin{array}{rll} (f+g)(x) &= f(x)+g(x) & < |-2| \\ h(x) &= (-\sin^2 x+\cos^2 x)+2\sin^2 x & < 2 \\ h(x) &= \sin^2 x+\cos^2 x & < 2 \\ h(x) &= 1 & < 2 \end{array} \tag{5.1}$$

Untuk menuliskan penurunan persamaan dan pertidaksamaan di atas, dapat digunakan kode di bawah ini:

```
\begin{equation}
\begin{array}{rll}
(f+g)(x)&=f(x)+g(x)&<|-2 |\\
h(x)&=(-\sin^2x+\cos^2x)+2\sin^2x&<2\\
h(x)&=\sin^2x+\cos^2x&<2\\
h(x)&=1&<2
\end{array}
\end{equation}
```

Dari contoh di atas, perhatikan bahwa kita telah menggunakan *environment* `equation` untuk membuat mode *displaymath* dengan nomor persamaan. Kita bisa saja membuat mode *displaymath* dengan cara yang lain atau memilih menggunakan mode *inline*. Perhatikan juga bahwa empat baris persamaan dan pertidaksamaan di atas hanya diberi satu buah nomor persamaan, sebagaimana pada *environment* `equation` pada umumnya. Ini berbeda dengan *environment* `align` yang bisa memberi nomor persamaan di setiap baris.

Bab 6

Menggambar dengan Ti*k*Z

Jangan dipahami bahwa kita akan menggambar objek-objek rumit dan detail seperti melukis wajah, membuat karikatur, atau menggambar pemandangan! Meskipun, jika kita ternyata adalah orang yang sedemikian kreatif, hal itu mungkin saja dilakukan. Unsur dasar gambar di antaranya adalah titik, garis atau kurva, bidang, dan warna. Berkutat di seputar itu saja kurang lebih pembahasan kita.

Untuk keperluan menggambar, dibutuhkan *package* khusus bernama `tikz`. *Package* ini cukup *powerfull* untuk membuat gambar-gambar matematis yang memerlukan ketepatan koordinat dalam pembuatannya. Akan tetapi pembahasan yang rinci dan lengkap akan sangat panjang dan pasti **tidak bisa disajikan seluruhnya** di bab ini. Bagi yang penasaran dan ingin tahu lebih banyak, silakan belajar juga dari sumber lain, terutama yang khusus membicarakan *package* `tikz`.

Pertama sekali agar perintah-perintah bawaan `tikz` dapat dimengerti, di *preamble* tuliskan

```
\usepackage{tikz}
```

Selanjutnya di dalam dokumen, setiap perintah untuk membuat ob-

jek (unsur-unsur gambar berupa garis, bidang, atau yang lain) akan dituliskan di dalam *environment* `tikzpicture`. Penggunaan *environment* sebenarnya bukanlah satu-satunya pilihan. Kita juga bisa memasukkan perintah untuk membuat objek setelah perintah `\tikz`. Akan tetapi pembuatan objek dengan didahului perintah `\tikz` akan efektif jika kita hanya membuat (menggambar) satu buah objek dalam ruang atau kanvas. Secara umum untuk menggambar banyak objek dalam satu kanvas, *environment* `tikzpicture` perlu digunakan.

6.1 Kanvas dan Sistem Koordinat

Pertama kita imajinasikan dulu sebuah objek "hayal" bernama kanvas. Istilah "kanvas" dalam penjelasan tentang Ti*k*Z mungkin hanya digunakan dalam buku ini dan tidak di tempat yang lain. Penggunaan istilah ini semata hanya cara penulis untuk memudahkan pembaca dalam memahami. Kanvas adalah ruang/permukaan dua dimensi tempat kita menggambar. Di ruang itu kita bisa menerapkan sistem koordinat yang berarti setiap titik di kanvas akan ditandai dengan dua bilangan. Dengan sistem koordinat cartesius, kita biasa menuliskannya (x, y) dengan x dan y keduanya bilangan. Kita juga bisa menggunakan sistem koordinat polar tetapi dalam praktiknya nanti sistem koordinat cartesius akan lebih banyak digunakan.

Satu buah *environment* `tikzpicture` yang nanti kita buat (yakni `\begin{tikzpicture}...\end{tikzpicture}`) sama maknanya dengan satu buah kanvas. Jika kita menamai titik berkoordinat $(0, 0)$ dengan A dan titik berkoordinat $(1, 0)$ dengan B (menggunakan sistem koordinat cartesius) di kanvas yang sama, titik B pasti berada di sebelah kanan titik A. Jika titik A dan B dihubungkan, akan didapatkan sebuah ruas garis lurus mendatar sepanjang 1 satuan.

Pertanyaan yang boleh jadi muncul di benak kita, "Di mana letak kanvas dalam dokumen kita jika nanti kita membuat kanvas?" Pertanyaan berikutnya, "Di mana pangkal koordinat dalam kanvas kita itu, apakah di tengah-tengah kanvas, pojok kiri bawah, atau di mana?" Jawaban bagi pertanyaan-pertanyaan ini adalah bahwa LaTeX akan l**ebih mengutamakan letak objek** yang kita gam-

bar dalam posisinya di dokumen dibandingkan dengan letak kanvas. Misalkan kita mengatur gambar agar berada di tengah-tengah dokumen, kemudian kita menggambar ruas garis AB seperti di atas, ruas garis yang muncul pasti berada di tengah-tengah dokumen. Jika titik A dan B koordinatnya diubah, misalkan $(1, 0)$ dan $(2, 0)$, gambar beserta letaknya (di dokumen) yang dihasilkan tetap sama karena sama-sama ruas garis mendatar sepanjang 1 satuan. Letak kanvas dan pangkal sistem koordinatnya akan menyesuaikan.

6.2 Penulisan Koordinat

Seperti telah dibicarakan sebelumnya, titik-titik di dalam kanvas ditandai dengan koordinat yang terdiri dari dua buah bilangan. Di sini, penulisan koordinat bagi titik dengan sistem koordinat cartesius menggunakan dua bilangan yang dipisahkan oleh koma (,) dan diapit oleh kurung. Secara *default*, jika tidak disebutkan satuannya, satuan yang digunakan adalah cm. Sebagai contoh, koordinat `(1,2)` berarti 1 cm arah mendatar (sumbu x) dan 2 cm arah vertikal (sumbu y) dari pangkal koordinat. Kita bisa menuliskan lebih lengkap dengan satuan, yakni `(1cm,2cm)`. Satuan lain boleh pula digunakan, misal mm, pt, di, ex dan in. Penulisan koordinat bagi titik dengan sistem koordinat polar menggunakan dua bilangan yang dipisahkan oleh titik dua (:) dan diapit oleh kurung. Secara *default*, satuan bagi bilangan pertama adalah derajat dan satuan bagi bilangan kedua adalah cm[1]. Satuan-satuan lain bisa juga digunakan dengan cara menuliskan (secara eksplisit) satuan-satuan itu.

Bilangan dalam koordinat boleh desimal, misal `(1.5,2.3)`. Jika bilangan sebelah kiri atau sebelah kanan titik adalah nol saja (atau disetai tanda minus) maka nol boleh tidak dituliskan. Misal, `(.,-.5)` sama maknanya dengan `(0.0,-0.5)`, tentu saja juga sama maknanya dengan `(0,-0.5)`.

[1] Urutannya memang terbalik jika dibandingkan kebiasaan menggunakan koordinat polar di matematika yang mana sudut biasanya ditempatkan pada urutan kedua.

6.3 Penamaan Titik dan Pelabelan

Setiap titik (yang sejatinya sudah ditandai oleh koordinat) dapat ditandai lagi dengan sesuatu yang lain. Misal, titik dengan koordinat `(1,2)` dapat ditandai (dinamai) lagi dengan `(A)`. Dalam memasukkan perintah nanti, `(1,2)` dapat diganti penulisannya dengan `(A)`. Cara menandai seperti ini dilakukan dengan menuliskan perintah berikut (di dalam *environment* `tikzpicture`):

```
\coordinate (A) at (1,2);
```

Jangan lupa menuliskan titik koma (;) di akhir. **Ini wajib!**

Titik dapat diberi label (berupa teks) di dalam dokumen dengan menyertakan *optional argument* bagi perintah `\coordinate`, yakni `[label=letak label:nama label]`[2]. Ganti `letak label` dengan `above, below, left, right, above left, above right, below left, below right,` atau `center` yang menandakan letak label ada di sebelah atas, bawah, kiri, kanan, atas kiri, atas kanan, bawah kiri, bawah kanan, atau tepat berada di titik itu! Ganti pula nama label dengan nama titik atau keterangan yang lain! Sering kali nama label berupa teks matematis yang penulisannya diapit oleh `$...$`. Label bagi sebuah titik boleh lebih dari satu. Jika pemberian label lebih dari satu, penulisannya dipisahkan dengan koma (,).

Contoh:
Tuliskan kode berikut!

```
\begin{tikzpicture}
\coordinate[label=above:$A$] (A) at (1,2);
\end{tikzpicture}
```

Luaran yang dihasilkan adalah A . Huruf A ini sebenarnya ada di atas titik berkoordinat (1,2), hanya saja titiknya tidak digambarkan. Supaya jelas keberadaan titiknya, dapat ditambahkan label

[2]Penggunaan istilah "label" di sini berbeda makna dengan istilah "label" di bab-bab sebelumnya (yang ditulis dengan perintah `\label` dan dipanggil dengan `\ref`)

baru berupa *bullet* (•) yang diletakkan tepat di titiknya. Tuliskan kode berikut!

```
\begin{tikzpicture}
\coordinate[label=above:$A$,label=center:$\bullet$] (A)
at (1,2);
\end{tikzpicture}
```

dan hasilnya menjadi 

.

Dari contoh terakhir kita telah menggambar sebuah titik beserta pelabelannya. Titik itu berada di koordinat (1,2). "Di mana letak titik pangkal (0,0)?" Yang jelas pasti ada di sebelah kiri bawah titik itu, tetapi itu tidak penting. LaTeX memilih untuk menempatkan titik itu setelah teks terakhir yang kita tulis. Hasilnya tampil setelah teks karena kita belum mengatur misal agar gambar diletakkan di baris terpisah dengan posisi di tengah-tengah.

6.4 Mengatur Letak Gambar

Untuk mengatur letak gambar yang kita buat, dibutuhkan bantuan *package* `graphicx` dan *environment* `figure`. *Environment* `tikzpicture` beserta isinya dibuat di dalam *environment* `figure`. Selanjutnya cara mengatur letak gambarnya adalah sama seperti yang telah dibahas di subbab 4.2.

Cara menggambar objek belum dibahas hingga bagian ini (selain titik). Sebagai gambaran saja, bermodalkan pembuatan label titik yang sudah dibahas di subbab 6.3, cukup kiranya digunakan untuk menunjukkan cara meletakkan gambar. Perhatikan contoh berikut!

Contoh:
Tuliskan kode di bawah ini!

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
```

```
\coordinate[label=above:$A$,label=center:$\bullet$] (A)
at (1,2);
\end{tikzpicture}
\end{figure}
```

Hasilnya menjadi

Penggunaan *environment* `figure` selain bermanfaat untuk mengatur peletakan gambar, sebagaimana yang telah dibahas, juga dapat digunakan untuk menambahkan *caption* dan keperluan lain.

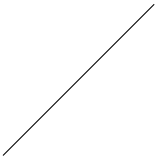
6.5 Menggambar Garis dan Poligon

Perintah untuk menggambar ruas garis lurus yang menghubungkan dua buah titik dari (koordinat 1) ke (koordinat 2) adalah

```
\draw (koordinat 1)--(koordinat 2);
```

Sekali lagi, **jangan lupa menuliskan titik koma (;)** di akhir!

Contoh:
Gambar ruas garis lurus (gambar 6.1), dibuat dengan menuliskan



Gambar 6.1: Ruas garis lurus

kode (*environment* `figure` di sini ikut digunakan agar gambar bisa diletakkan di tengah dengan disertai *caption*)

```
\begin{figure}[h!]
```

```
\centering
\begin{tikzpicture}
\draw (0,0)--(2,2);
\end{tikzpicture}
\caption{Ruas garis lurus}
\end{figure}
```

Agar lebih mahir dengan beragam cara penulisan kode, berikut disajikan dua kode lagi yang berbeda dengan hasil yang sama.

Kode lain (1) :

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (1,1)--(3,3);
\end{tikzpicture}
\caption{Ruas garis lurus}
\end{figure}
```

Kode lain (2) :

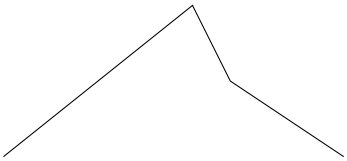
```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\coordinate (P) at (0.0);
\coordinate (Q) at (2.2);
\draw (P)--(Q);
\end{tikzpicture}
\caption{Ruas garis lurus}
\end{figure}
```

Jika terdapat sejumlah titik (lebih dari dua) yang kita hubungkan dengan garis-garis lurus maka dapat terbentuk poligon. Perintah untuk membuat poligon adalah

```
\draw (koordinat 1)--(koordinat 2)--...-- (koordinat n);
```

(titik-titik berarti "dan seterusnya").

Contoh:
Poligon terbuka pada gambar 6.2 dapat dibuat (salah satunya) de-



Gambar 6.2: Poligon terbuka

ngan menuliskan kode

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (-.5,0)--(2,2)--(2.5,1)--(4,0);
\end{tikzpicture}
\caption{Poligon terbuka}
\end{figure}
```

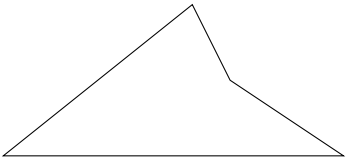
Poligon yang telah dibuat pada contoh di atas adalah poligon terbuka (titik awal dan akhir tidak tersambung). Untuk membuat poligon tertutup, ada dua cara yang mungkin ditempuh. Pertama adalah dengan menambahkan satu titik terakhir berkoordinat sama dengan koordinat titik pertama. Kedua yaitu dengan menambahkan `cycle` sebagai pengganti titik terakhir (yang sama koordinatnya dengan titik pertama). Lebih jelasnya untuk cara kedua, perintahnya adalah sebagai berikut:

```
\draw (koordinat 1)-- ...--(koordinat n)--cycle;
```

Contoh:
Poligon tertutup pada gambar 6.3 dapat dibuat (salah satunya) dengan menuliskan kode

```
\begin{figure}[h!]
\centering
```

```
\begin{tikzpicture}
\draw (-.5,0)--(2,2)--(2.5,1)--(4,0)--cycle;
\end{tikzpicture}
\caption{Poligon tertutup}
\end{figure}
```



Gambar 6.3: Poligon tertutup

Setelah kita mengetahui cara membuat poligon tertutup, tentu saja kita bisa membuat bidang-bidang yang sisi-sisinya berupa garis-garis lurus, seperti segitiga, persegi panjang, jajar genjang, dan masih banyak lagi. Akan tetapi terdapat pula cara-cara khusus untuk membuat beberapa objek, di antaranya adalah persegi panjang (ada perintahnya tersendiri).

6.6 Modifikasi Gambar

Perintah `\draw` tidak hanya digunakan untuk menggambar garis dan poligon tetapi juga untuk menggambar beberapa objek yang akan dibicarakan di subbab-subbab berikutnya. Cara memodifikasi gambar yang dijelaskan pada bagian ini nantinya juga dapat diterapkan ke objek-objek tersebut.

Perintah `\draw` dapat disertai dengan *optional argument*. Banyak opsi untuk memodifikasi gambar yang dapat dituliskan di dalam *optional argument*. Sebagaimana biasa, masing-masing opsi dipisahkan oleh koma (,). Beberapa dari opsi-opsi itu antara lain:

1. **Pengaturan ketebalan garis.** Kode untuk mengatur ketebalan garis dari yang paling tipis hingga yang paling tebal yaitu `ultra thin, very thin, thin, semithick, thick,`

`very thick`, dan `ultra thick`. Selain dengan ini, dapat juga menggunakan `line width=...` Titik-titik diganti dengan ketebalan garis dan bisa disertai satuan. Jika tanpa satuan, misal `line width=3` maka otomatis dianggap satuannya adalah `pt`.

2. **Pembuatan garis putus-putus**. Tampilan garis dapat dibuat putus-putus yang merupakan gabungan strip (-), titik, ataupun selang-seling strip dan titik. Kode yang digunakan berturut-turut adalah `dashed`, `dottted`, dan `dash dot`.

3. **Pembuatan garis rangkap**. Garis rangkap dibuat dengan kode `double`.

4. **Pengaturan ujung garis.** Ujung garis (awal dan/atau akhir) dapat diberi anak panah atau yang semacamnya. Kode yang digunakan adalah `->`, `<-`, `-|`, `|-`, `->>`, `<<-`, `-|>`, `<|-`, `<->`, `|-|`, `|->`, `<-|`, `-stealth`, `-latex`, dan beberapa lagi yang lain. Untuk poligon tertutup yang penulisan kodenya menggunakan `cycle`, pengaturan ujung garis tidak berlaku.

5. **Pewarnaan garis**. Garis dapat diubah warnanya dengan menuliskan `color=warna` atau langsung menyebutkan `warna` (ganti `warna` sesuai dengan nama-nama warna yang diinginkan!). Warna dapat juga dibuat lebih cerah atau lebih gelap dari warna aslinya dengan menyertakan tanda seru (`!`) dan diikuti bilangan (menyatakan "sekian" persen lebih cerah atau lebih gelap).

6. **Pewarnaan isi**. Wilayah yang dibatasi kurva dapat diwarnai dengan kode `fill=warna`. Ganti `warna` sesuai warna yang diinginkan (dapat pula dibuat menjadi lebih cerah atau lebih gelap).

7. **Pemberian *pattern* (pola)**. Untuk memberi *pattern* pada wilayah yang dibatasi kurva, terlebih dahulu di bagian *preamble* perlu dituliskan \usetikzlibrary{`patterns`}. Setelah itu untuk membuat *pattern* digunakan kode `pattern=jenis pattern`. Ganti `jenis pattern` dengan salah satu dari beberapa jenis *pattern* berikut: `dots`, `grid`, `horizontal lines`,

`vertical lines`, `north east lines`, `north west lines`, `crosshatch`, `crosshatch dots`, `fivepointed stars`, dan `sixpointed stars`. *Pattern* dapat diberi warna dengan menuliskan `pattern=warna pattern`. Ganti `warna pattern` dengan warna tertentu!

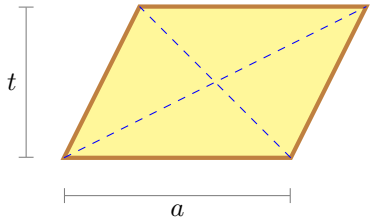
8. **Penghilangan sudut kurva (*rounded*)**. Sudut kurva dapat dihilangkan (menjadi *rounded*) dengan menuliskan kode `rounded corners=bilangan`. Semakin besar `bilangan` berarti semakin besar bagian lingkaran yang menggantikan sudut atau hasilnya menjadi "semakin licin/halus".

9. **Pemutaran gambar**. Gambar dapat diputar dengan kode `rotate=sudut putar`. Ganti `sudut putar` dengan besar sudut putar yang diinginkan (dalam derajat).

10. **Skala**. Skala (perbesaran gambar) dapat diatur dengan menuliskan kode `skala=besar skala`.

Contoh:
Dengan hanya bermodalkan pengetahuan tentang perintah menggambar garis dan poligon yang dimodifikasi serta pelabelan titik, kita akan coba membuat dua macam gambar berikut:

1. Jajar genjang berwarna pada gambar 6.4 dibuat dengan me-



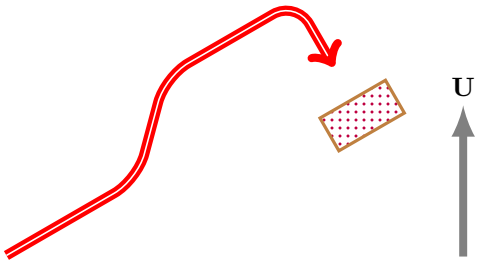
Gambar 6.4: Jajar genjang

nuliskan kode

```
\begin{figure}[h!]
```

```
\centering
\begin{tikzpicture}
\draw[brown,ultra thick,fill=yellow!50] (0,0)--(3,0)
--(4,2)--(1,2)--cycle;
\draw[blue,dashed] (0,0)--(4,2);
\draw[blue,dashed] (3,0)--(1,2);
\draw[|-|,gray] (-.5,0)--(-.5,2);
\draw[|-|,gray] (0,-.5)--(3,-.5);
\coordinate[label=left:$t$] (t) at (-.5,1);
\coordinate[label=below:$a$] (a) at (1.5,-.5);
\end{tikzpicture}
\caption{Jajar genjang}
\end{figure}
```

2. Perhatikan denah jalan pada gambar 6.5! Gambar ini dibuat



Gambar 6.5: Denah jalan

dengan menuliskan kode berikut:

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw[rounded corners=10,double,red,ultra thick,
->,rotate=30] (0,0)--(2,0)--(3,1)--(5,1)--(5,0);
\draw[brown,very thick, pattern=dots,pattern color
=purple,rotate=30] (4.5,-.5)--(5.5,-.5)--(5.5,-1)
--(4.5,-1)--cycle;
\endtikzpicture
```

```
\hspace{0.25cm}
\begin{tikzpicture}
\draw[-latex,line width=3pt,gray](0,0)--(0,2);
\coordinate[label=above:\textbf{U}] (U) at (0,2);
\end{tikzpicture}
\caption{Denah jalan}
\end{figure}
```

Perhatikan bahwa di sini kita menggunakan dua kanvas (*environment* `tikzpicture`)!

6.7 Menggambar Persegi Panjang

Agar lebih banyak bekal kita menggambar, cara membuat objek-objek lain perlu juga dikenalkan. Perintah untuk menggambar persegi panjang dengan salah satu titik sudut berada di (koordinat 1) dan satu titik sudut lagi yang terjauh di (koordinat 2) dengan arah (koordinat 1) ke (koordinat 2) diagonal adalah

```
\draw (koordinat 1)rectangle(koordinat 2);
```

Contoh:
Perhatikan persegi panjang pada gambar 6.6! Persegi panjang ini

Gambar 6.6: Persegi panjang

dibuat dengan menuliskan kode berikut:

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (1,2)rectangle(5,4);
```

```
\end{tikzpicture}
\caption{Persegi panjang}
\end{figure}
```

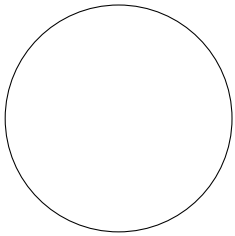
6.8 Menggambar Lingkaran

Perintah untuk menggambar lingkaran berpusat di (koordinat pusat) dengan besar jari-jari tertentu adalah

```
\draw (koordinat pusat)circle(jari-jari);
```

Jari-jari lingkaran jika tidak disebutkan satuannya, otomatis satuannya adalah cm.

Contoh:
Perhatikan lingkaran pada gambar 6.7! Lingkaran ini dibuat dengan



Gambar 6.7: Lingkaran

menuliskan kode berikut:

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (1,2)circle(1.5cm);
\end{tikzpicture}
\caption{Lingkaran}
\end{figure}
```

6.9 Menggambar Elips

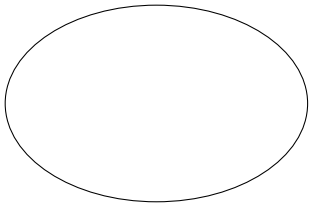
Perintah untuk menggambar elips yang berpusat di (koordinat pusat) dengan besar jari-jari horisontal tertentu dan besar jari-jari vertikal tertentu adalah

```
\draw (koordinat pusat)ellipse(jari-jari horisontal and
jari-jari vertikal);
```

Jari-jari horisontal dan vertikal jika tidak disebutkan satuannya, otomatis satuannya adalah cm. Perhatikan bahwa kita telah menggunakan istilah jari-jari horisontal dan vertikal, bukan jari-jari mayor dan minor! Ini karena yang menjadi jari-jari mayor ataupun minor boleh saja jari-jari horisontal ataupun vertikal. Perhatikan pula penggunaan `and` untuk memisahkan jari-jari horisontal dan vertikal!

Contoh:
Perhatikan elips pada gambar 6.8! Elips ini dibuat dengan kode berikut:

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (1,1)ellipse(2cm and 1.3cm);
\end{tikzpicture}
\caption{Elips}
\end{figure}
```



Gambar 6.8: Elips

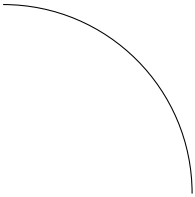
6.10 Menggambar Busur Lingkaran

Perintah untuk menggambar busur lingkaran yang mulai digambar dari (koordinat awal) dan dari sudut[3] 1 ke sudut 2 pada lingkaran dengan jari-jari tertentu adalah

```
\draw (koordinat awal)arc(sudut 1:sudut 2:jari-jari);
```

Sudut jika tidak disebutkan satuannya maka satuannya adalah derajat. Perhatikan penggunaan titik dua (:)!

Contoh:
Busur lingkaran (berpusat di (0,0)) pada gambar 6.9 dibuat dengan



Gambar 6.9: Busur lingkaran

menuliskan kode berikut ini:

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (2.5,0)arc(0:90:2.5cm);
\end{tikzpicture}
\caption{Busur lingkaran}
\end{figure}
```

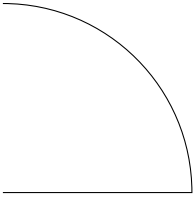
Busur lingkaran adalah kurva sebagaimana garis tetapi yang

[3]Besarnya sudut diukur dari garis horisontal yang berpangkal di pusat lingkaran ke kanan dan bernilai positif jika berlawanan dengan arah putar jarum jam.

membedakannya adalah busur lingkaran punya kelengkungan. Kita telah bisa membuat poligon dengan menghubung-hubungkan garis. Demikian halnya kita juga bisa membuat objek hasil gabungan dari kurva-kurva yang salah satu bagiannya adalah busur lingkaran, termasuk di dalamnya dapat pula kita menggunakan `cycle` untuk membuat kurva tertutup.

Contoh:

1. Gambar 6.10, gabungan sebuah ruas garis lurus dan busur



Gambar 6.10: Objek hasil gabungan garis dan busur lingkaran

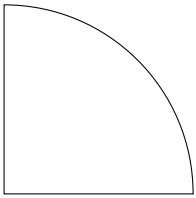
lingkaran, dibuat menggunakan kode

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (0,0)--(2.5,0)arc(0:90:2.5cm);
\end{tikzpicture}
\caption{Objek hasil gabungan garis dan busur
lingkaran}
\end{figure}
```

2. Juring lingkaran pada gambar 6.11 dibuat menggunakan kode berikut:

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (0,0)--(2.5,0)arc(0:90:2.5cm)--cycle;
\end{tikzpicture}
```

```
\caption{Juring lingkaran}
\end{figure}
```



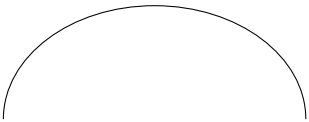
Gambar 6.11: Juring lingkaran

6.11 Menggambar Busur Elips

Perintah untuk menggambar busur elips yang mulai digambar dari (koordinat awal) dan dari sudut 1 ke sudut 2 pada elips dengan jari-jari horisontal dan jari-jari vertikal tertentu adalah

```
\draw (koordinat awal)arc(sudut 1:sudut 2:jari-jari
horisontal and jari-jari vertikal);
```

Contoh:
Busur elips (setengah elips) pada gambar 6.12 dibuat dengan meng-



Gambar 6.12: Busur elips

gunakan kode berikut:

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (2,0)arc(0:180:2 and 1.5);
```

```
\end{tikzpicture}
\caption{Busur elips}
\end{figure}
```

Seperti halnya busur lingkaran, busur elips juga merupakan kurva dan dapat dihubung-hubungkan dengan kurva-kurva lain (membentuk suatu kurva baru hasil gabungan beberapa kurva) dalam satu perintah `\draw`.

6.12 Menambahkan Teks pada Gambar

Secara tidak langsung kita telah belajar satu cara menambahkan teks pada gambar di subbab 6.3. Akan tetapi apa yang sesungguhnya dipelajari pada subbab itu adalah cara menamai titik, hanya saja "bonusnya" kita bisa menambahkan lebel dan label itu akan muncul di dokumen sebagai teks.

Cara lain menambahkan teks pada (koordinat) dari suatu titik tertentu adalah dengan menuliskan perintah

```
\node at (koordinat) {teks};
```

Teks juga dapat diletakkan tidak tepat di koordinatnya tetapi di atasnya, atas kanannya, dan seterusnya dengan cara mengganti titik-titik di bagian *optional argument* pada perintah berikut

```
\node[...]  at (koordinat) {teks};
```

dengan `above, above right,` dan lain sebagainya (sebagaimana *optional argument* pada perintah `\coordinate`, kecuali `center`).

Contoh:
Luaran dari kode berikut ini,

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw (0,0)--(0,2);
\node[above] at (0,2) P;
```

```
\endtikzpicture
\captionTeks di atas garis
\endfigure
```

adalah sebagaimana ditunjukkan oleh gambar 6.13. Terlihat bahwa letak P di atas titik yang ditempati oleh ujung atas garis.

Gambar 6.13: Teks di atas ruas garis vertikal

Jika dikehendaki, penamaan titik dengan koordinat sesuai letak teks (bukan titik di (koordinat) yang telah dibicarakan) juga dapat dilakukan di sini dengan menambahkan `(nama titik)`, misal `(X)`, sebelum `at`. Lebih jelasnya, pada perintah

```
\node[above] (X) at (0,0) {teks};
```

`X` menamai titik dengan koordinat di atas (0,0) yang merupakan tempat "teks" berada.

Selanjutnya pada setiap penggambaran objek yang melibatkan koordinat-koordinat, pada masing-masing koordinat dapat disertai dengan teks yang perintahnya tidak terpisah dari perintah untuk menggambar objek itu. Caranya adalah dengan menambahkan

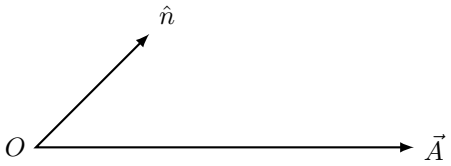
```
node[letak teks] {teks}
```

setelah menuliskan koordinat. Seperti sebelumnya, `letak teks` diganti dengan `above, below,` dan lain-lain.

Contoh:

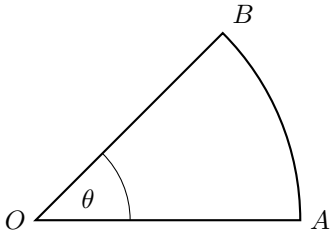
1. Perhatikan gambar 6.14! Gambar ini dibuat dengan menggunakan kode

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw[latex-latex,very thick] (5,0)node[right]
{$\vec{A}$}--(0,0)node[left]{$O$}--(1.5,1.5)
node[above right]{$\hat{n}$};
\end{tikzpicture}
\caption{Dua buah vektor}
\end{figure}
```



Gambar 6.14: Dua buah vektor

2. Untuk membuat gambar 6.15, sebuah juring lingkaran dengan



Gambar 6.15: Juring lingkaran dengan sudut θ

sudut θ, kode yang dituliskan adalah

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\draw[thick] (0,0)node[left]{$O$}--(3.5,0)
node[right]{$A$}arc(0:45:3.5)node[above right]
```

```
{$B$}--cycle;
\draw[thin] (1.25,0)arc(0:45:1.25);
\node at (22.5:.75){$\theta$};
\end{tikzpicture}
\caption{Juring lingkaran dengan sudut $\theta$}
\end{figure}
```

6.13 Sekilas Mengenal PGFPlots

Teramat kompleks dan detail cara-cara menggambar dengan LaTeX jika hendak dijelaskan seluruhnya. Bagian ini hanya digunakan untuk mengenalkan PGFPlots kepada pembaca. Sekali lagi, sekedar agar pembaca mengenal, bukan untuk menjelaskan PGFPlots secara rinci. Diharapkan dari mengenal, apabila pembaca penasaran dan ingin tahu lebih lanjut, pembaca dapat mencari sumber-sumber lain yang lebih memadai sebagai bahan belajar.

PGFPlots dapat digunakan untuk mengeplot grafik dari suatu persamaan fungsi matematis. PGFPlots dapat dijalankan salah satunya ketika kita telah menggunakan *package* `tikz`. Untuk dapat mengeplot grafik, pada *preamble* perlu dituliskan

```
\usepackage{pgfplots}
```

Selanjutnya untuk mengatur lebar gambar grafik yang dihasilkan, masih di *preamble* perlu dituliskan lagi

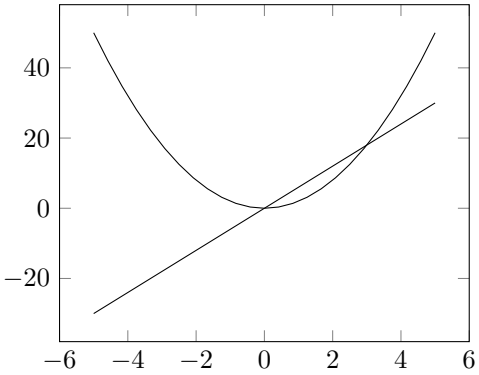
```
\pgfplotsset{width=...}
```

Titik-titik diganti bilangan beserta satuannya yang menyatakan lebar gambar, misal `8cm`.

Di dalam dokumen, perintah mengeplot grafik dituliskan di dalam *environment* `axis`, yakni `\begin{axis}...\end{axis}`. *Environment* ini dibuat di dalam *environment* `tikzpicture`. Perintah untuk mengeplot grafik adalah `\addplot{...};`. Perintah semacam ini dapat dituliskan lebih dari satu kali untuk keperluan mengeplot beberapa grafik sekaligus. Titik-titik diganti dengan fungsi yang

ingin diplot. Apabila persamaan fungsi itu adalah $y = f(x)$, yang harus dituliskan untuk mengganti titik-titik adalah bentuk eksplisit dari $f(x)$. Misal, $y = 2x + 1$, yang harus ditulis adalah `2*x+1`. Notasi `*` menyatakan kali. Ada aturan-aturan untuk menulis fungsi (kali ditulis dengan `*`, pangkat ditulis dengan `^`, akar ditulis dengan `sqrt(...)`, dan lain-lain) tetapi perihal ini **tidak akan dibicarakan detail** di sini.

Contoh:
Pastikan perintah-perintah yang harus ada di *preamble* sudah dituliskan dan atur lebar gambar `8cm`. Selanjutnya kita akan coba plot



Gambar 6.16: Grafik parabola dan garis

grafik dari fungsi $y = 2x^2$ dan $y = 6x$. Tuliskan kode berikut!

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\begin{axis}
\addplot[black]{2*x^2};
\addplot[black]{6*x};
\end{axis}
\end{tikzpicture}
\caption{Grafik parabola dan garis}
```

```
\end{figure}
```

Hasil yang didapatkan adalah seperti pada gambar 6.16.

Cara mengeplot grafik seperti pada contoh di atas tidak mengikutsertakan penggambaran sumbu horisontal dan vertikal yang berpotongan di titik (0,0). Agar sumbu (berikut nama-nama sumbunya) ikut digambarkan dan kedua sumbu itu berpotongan di titik (0,0), *optional argument* dari *environment* `axis` berikut ini perlu ditambahkan:

```
[axis lines=middle,
xlabel={...}, ylabel={...}]
```

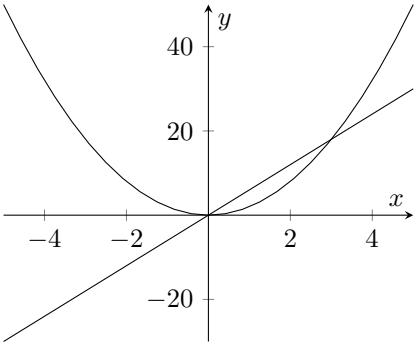
Titik-titik diisi nama masing-masing sumbu (sumbu horisontal dan sumbu vertikal).

Contoh:
Tuliskan kode berikut!

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}
\begin{axis} [axis lines=middle,
xlabel={$x$}, ylabel={$y$}]
\addplot[black]{2*x^2};
\addplot[black]{6*x};
\end{axis}
\end{tikzpicture}
\caption{Grafik parabola dan garis dengan sumbu}
\end{figure}
```

Luaran yang didapatkan adalah seperti pada gambar 6.17. Perhatikan bahwa dua sumbu (x dan y) telah tergambar dan berpotongan di $(0, 0)$.

Sebatas ini saja paparan tentang PGFPlots di buku ini dan silakan



Gambar 6.17: Grafik parabola dan garis dengan sumbu

dilanjutkan belajar mandiri dari sumber lain (yang lebih lengkap penjelasannya) jika tertarik!

Bab 7

Membuat Presentasi dengan Beamer

Dokumen luaran dari LaTeX adalah berupa file pdf, tidak terkecuali dalam pembuatan media presentasi. Mungkin orang bertanya, "Apa menariknya presentasi dengan pdf?" Benar bahwa pdf tidak menampilkan gambar bergerak, video, ataupun menghadirkan suara atau musik. Akan tetapi di forum seperti apa presentasi itu ditampilkan turut menentukan media apa yang lebih baik untuk digunakan. Di dalam presentasi ilmiah yang perlu menampilkan banyak persamaan matematis, musik bukan sesuatu yang penting. Yang lebih utama dibutuhkan adalah kemampuan media presentasi menyajikan penulisan persamaan-persamaan matematis yang baik, juga file yang tidak rusak saat dibuka di sembarang komputer. Kualitas file sekelas ini dimiliki oleh pdf.

7.1 Sekilas tentang Beamer dan Efek

Terdapat beberapa kelas dokumen di LaTeX yang dikhususkan untuk pembuatan media presentasi, salah satunya adalah `beamer`. Beberapa yang lain yaitu `slides, seminar, prosper,` dan `powerdot`. *Slides* oleh `beamer` disebut dengan istilah *frames*. File luaran yang dihasilkan adalah berupa pdf dengan banyak halaman dan (tentu

saja) banyak *frames*. Di dalam menampilkan presentasi, setiap halaman ditampilkan bergantian, bukan beberapa halaman tampil di saat bersamaan.

"Bagaimana dengan efek?" Efek yang kita jumpai di media presentasi lain "dibentuk" dari perubahan tampilan. Bahkan, gambar bergerak dengan ekstensi .gif yang biasa kita temui pun sesungguhnya adalah gambar yang berganti-ganti. Jelaslah bahwa efek dan kesan bergerak dapat pula dimunculkan dari tampilan halaman pdf yang berganti-ganti. Bedanya, perubahan tampilan pada presentasi dengan menggunakan file pdf hanya terjadi saat pergantian halaman. Akibatnya, tidak ada perubahan-perubahan tampilan yang *smooth*.

7.2 Keuntungan Menggunakan Beamer

Beberapa keuntungan menggunakan `beamer` antara lain:

1. Perintah-perintah umum di LaTeX dimengerti oleh `beamer`. Perintah seperti `\section,\subsection,` dan seterusnya masih dapat digunakan. Perintah untuk membuat daftar penomoran, menuliskan teks matematis, menginput gambar, menambahkan tabel, keseluruhannya juga masih dapat digunakan.

2. Daftar isi dibuat secara otomatis semudah menuliskan perintah `\tableofcontents`.

3. File pdf yang dihasilkan dilengkapi dengan berbagai hyperlink. Kita bisa klik bagian tertentu di daftar isi untuk menuju ke suatu *frame* tertentu.

4. Pada `beamer` tersedia berbagai pilihan *themes* (tema-tema) dengan tampilan yang berbeda-beda sehingga kita tidak perlu mendesain manual. Akan tetapi jika kita ingin mendesain manual, hal ini sangat mungkin dilakukan karena perintah untuk menggambar dan sebagainya masih berfungsi. Pada masing-masing *theme* juga dapat diubah warnanya secara global, tampilan bagian dalam, tampilan bagian luar, dan jenis

font, sebab pada masing-masing tema itu terdapat tema-tema lain yang menyertainya.

5. Tersedia berbagai pilihan *overlay* dengan perintah-perintah khusus. *Overlay* dapat dipahami sebagai efek untuk memunculkan bagian-bagian *frame*.

7.3 Memulai Pembuatan Presentasi

Sebagaimana pembuatan dokumen pada umumnya, pertama kali yang harus dituliskan adalah kelas dokumen yang dalam hal ini adalah `beamer`. Tuliskan

```
\documentclass{beamer}
```

Selanjutnya di bagian *preamble* kita juga tuliskan berbagai *package* yang dibutuhkan. Misal,

```
\usepackage[bahasa]{babel}
\usepackage{amsmath}
\usepackage{graphicx}
```

Berikutnya, masih di *preamble*, kita bisa menuliskan tema yang akan digunakan. Perintahnya adalah

```
\usetheme{nama tema}
```

Jika perintah untuk memilih tema tidak dituliskan, tema yang digunakan adalah tema *default* dengan *background* putih. Beberapa `nama tema` yang bisa dipilih antara lain:

`singapore`, `cambridgeus`, `copenhagen`, `montpelier`, `antibes`, `marburg`, `madrid`, `limenau`, `juanlespins`, `berlin`, `paloalto`, `rochester`, `warsaw`, `hannover`, `berkeley`, dan `malmoe`.

Masih banyak tema lain tetapi tidak seluruhnya disebutkan di sini.

Setiap tema disertai dengan tema-tema yang mengikutinya, yaitu *color theme*, *inner theme*, *outer theme*, dan *font theme*. Jika

ingin menyertakan pilihan dari tema-tema yang mengikuti ini, tambahkan lagi pada *preamble*

```
\usecolortheme{tema warna}
\useinnertheme{tema dalam}
\useoutertheme{tema luar}
\usefonttheme{tema font}
```

Pilihan untuk `tema warna` antara lain:

`whale, orchid, dolphin, beetle, beaver,` dan `seahorse`.

Pilihan untuk `tema dalam` antara lain:

`circles, rounded, rectangles,` dan `inmargin`.

Pilihan untuk `tema luar` antara lain:

`smoothbars, infolines, sidebar, split,` dan `tree`.

Pilihan untuk `tema font` antara lain:

`serif, structurebold, structureitalicserif,` dan `structuresmallcapsserif`.

Berikutnya, sebelum masuk ke bagian isi dokumen, kita bisa menuliskan dulu judul dan beberapa yang menyertainya,

```
\title{Nama Judul}
\author{Pengarang}
\date{Tanggal}
```

Setelahnya kita masuk ke bagian isi dokumen. Tuliskan lebih dulu `\begin{document}...\end{document}`. Di dalamnya kita bisa membuat *frame* menggunakan *environment* `frame` dengan disertai `nama file` (tapi tidak wajib), yakni `\begin{frame}{nama frame}...\end{frame}`. Untuk menghasilkan banyak *frame*, kita harus membuat *environment* `frame` berkali-kali. Bagian inti dokumen kita menjadi

```
\begin{document}
\begin{frame}{nama frame}
...
\end{frame}
\begin{frame}{nama frame}
...
\end{frame}
...
dst.
\end{document}
```

Agar lebih mudah memahami, melalui contoh berikut ini kita akan melihat tampilan luaran yang dihasilkan dari pembuatan presentasi sederhana.

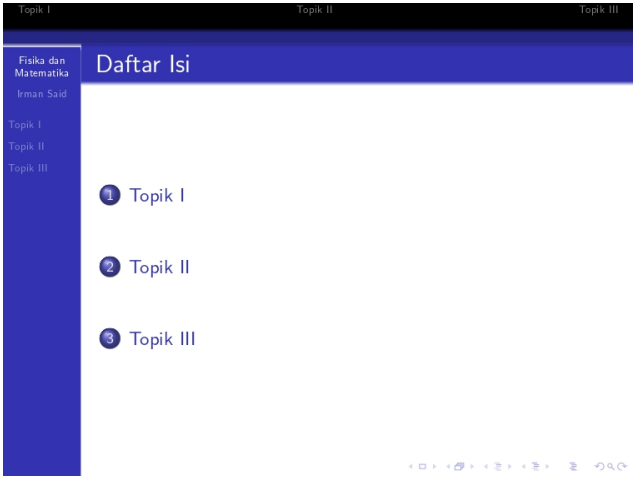
Contoh:
Kita akan mencoba membuat presentasi sederhana menggunakan kode berikut:

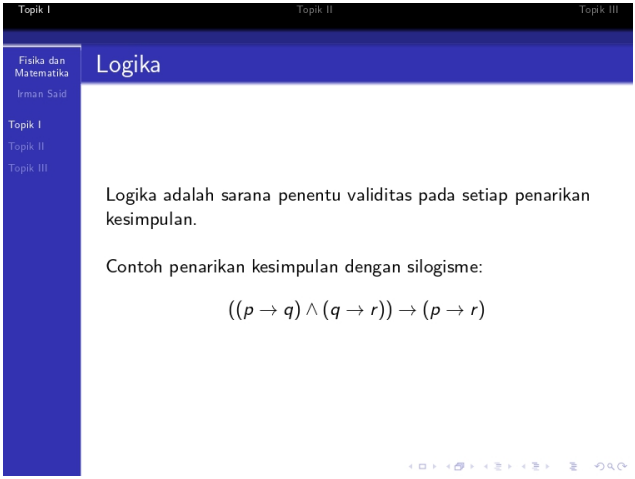
```
\documentclass{beamer}

\usepackage[bahasa]{babel}
\usepackage{amsmath}
\usepackage{graphicx}

\usetheme{berkeley}
\usecolortheme{orchid}
\useinnertheme{rounded}
\useoutertheme{smoothbars}

\title{Fisika dan Matematika}
\author{Irman Said}
\date{1 September 2021}

\begin{document}

\begin{frame}
```

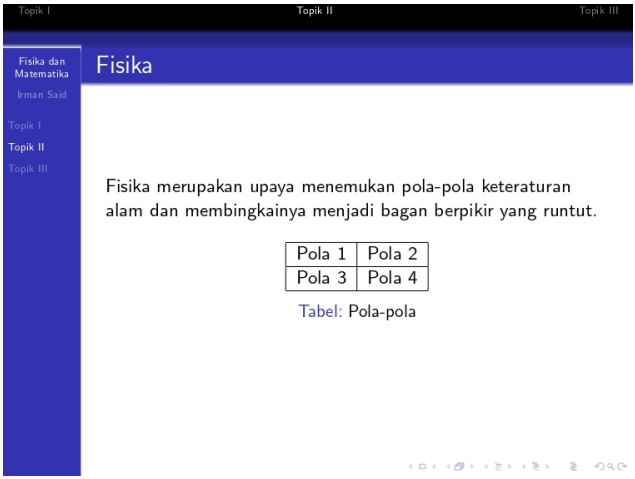
```
\maketitle
\end{frame}

\begin{frame}{Daftar Isi}
\tableofcontents
\end{frame}

\begin{frame}{Logika}
\section{Topik I}
Logika adalah sarana penentu validitas pada setiap
penarikan kesimpulan.\\[0.5cm]
Contoh penarikan kesimpulan dengan silogisme:
\[((p\to q)\wedge(q\to r))\to(p\to r)\]
\end{frame}

\begin{frame}{Fisika}
\section{Topik II}
Fisika merupakan upaya menemukan pola-pola keteraturan
alam dan membingkainya menjadi bagan berpikir yang
runtut.
\begin{table}
\centering
\begin{tabular}{|c|c|}
\hline
Pola 1&Pola 2\\
\hline
Pola 3&Pola 4\\
\hline
\end{tabular}
\caption{Pola-pola}
\end{table}
\end{frame}

\begin{frame}{Matematika}
\section{Topik III}
Matematika adalah bahasa bagi ilmu alam.  Pernyataan-
pernyataan matematis memiliki nilai kebenaran yang
didasarkan pada kebenaran aksioma-aksioma.
```

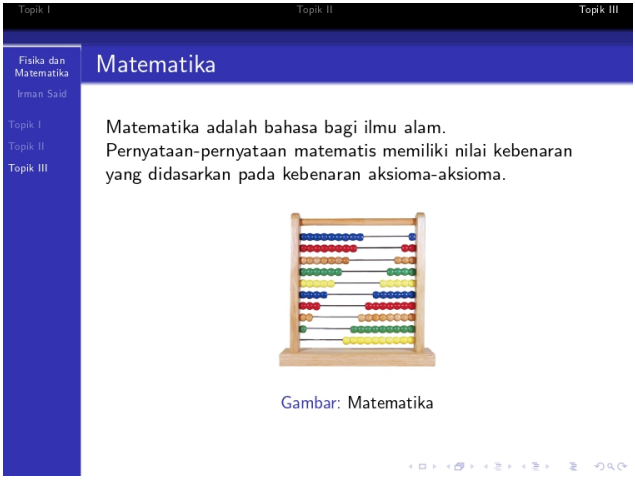
```
\begin{figure}
\centering
\includegraphics[scale=0.2]{Abacus}
\caption{Matematika}
\end{figure}
\end{frame}

\end{document}
```

Luaran dari kode di atas adalah seperti diperlihatkan pada gambar 7.1.


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Daftar Isi
1 Topik I
2 Topik II
3 Topik III


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Logika
Logika adalah sarana penentu validitas pada setiap penarikan kesimpulan.
Contoh penarikan kesimpulan dengan silogisme:
$((p \to q) \wedge (q \to r)) \to (p \to r)$





Gambar 7.1: Presentasi sederhana

Pada contoh di atas kita telah membuat presentasi yang menam-

pilkan satu *frame* berisi judul, satu *frame* berisi daftar isi, dan tiga *frame* untuk tiga *section*. Daftar isi dimunculkan secara otomatis berdasarkan semua *section* yang telah dibuat. Pada contoh di atas kita juga bisa melihat bahwa teks matematis, tabel, dan gambar dapat ditambahkan ke dokumen. Akan tetapi kita belum menyertakan efek. Beberapa subbab setelah ini akan membahas tentang *overlay*, yakni semacam efek untuk menampilkan isi *frame*.

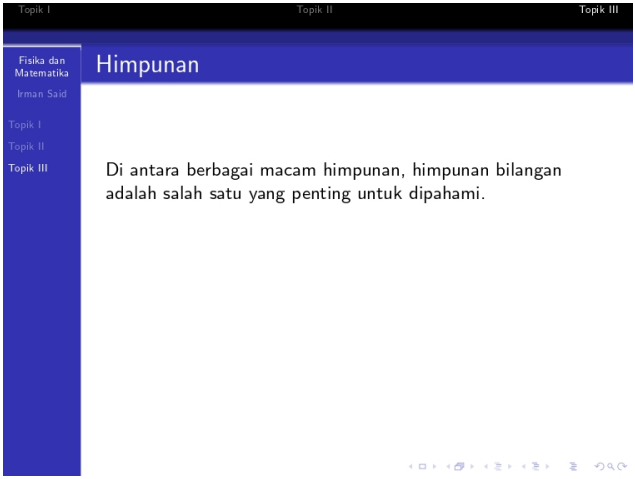
7.4 *Overlay Pause*

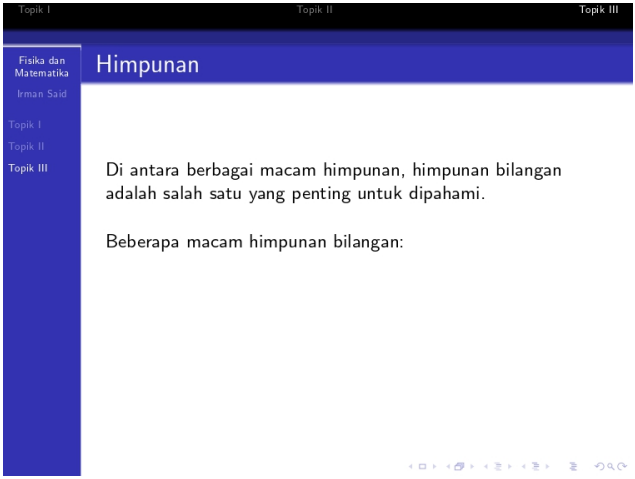
Pada saat menampilkan presentasi, bagian-bagian *frame* dapat dimunculkan satu demi satu. Caranya adalah dengan menambahkan perintah `\pause` pada bagian-bagian teks atau bagian-bagian*frame* yang menjadi batas-batas saat penampilan bertahap.

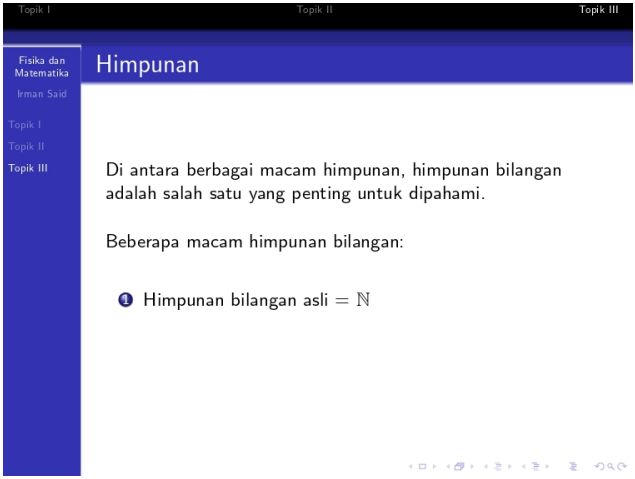
Contoh:
Dari kode yang telah dituliskan pada contoh sebelumnya, sebelum `\end{document}` tambahkan kode untuk *frame* baru berikut:

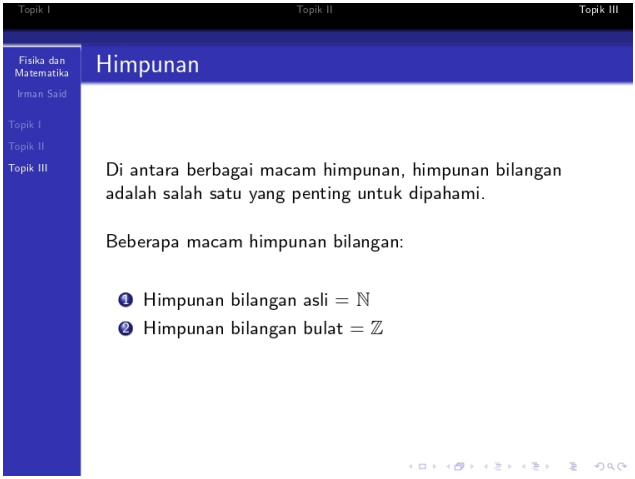
```
\begin{frame}{Himpunan}
Di antara berbagai macam himpunan, himpunan bilangan
adalah salah satu yang penting untuk dipahami.\pause
\\[0.5cm]
Beberapa macam himpunan bilangan:\pause
\\[0.5cm]
\begin{enumerate}
\item Himpunan bilangan asli = $\mathbb{N}$\pause
\item Himpunan bilangan bulat = $\mathbb{Z}$\pause
\item Himpunan bilangan riil = $\mathbb{R}$
\end{enumerate}
\end{frame}
```

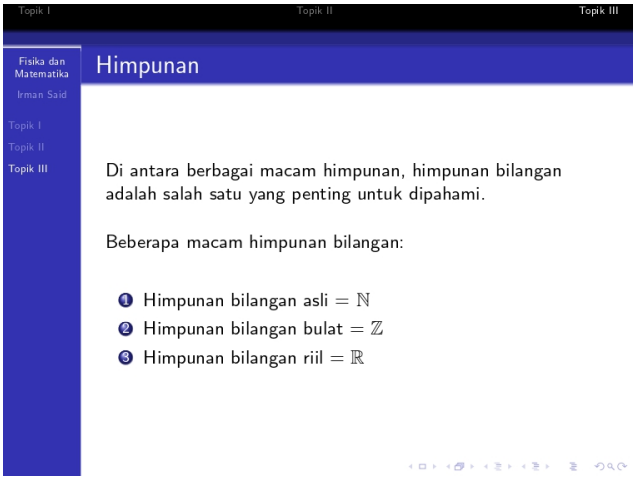
Hasil dari kode satu *frame* ini adalah lima halaman pdf seperti pada gambar 7.2.


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Himpunan
Di antara berbagai macam himpunan, himpunan bilangan adalah salah satu yang penting untuk dipahami.


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Himpunan
Di antara berbagai macam himpunan, himpunan bilangan adalah salah satu yang penting untuk dipahami.
Beberapa macam himpunan bilangan:


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Himpunan
Di antara berbagai macam himpunan, himpunan bilangan adalah salah satu yang penting untuk dipahami.
Beberapa macam himpunan bilangan:
1 Himpunan bilangan asli = $\mathbb{N}$


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Himpunan
Di antara berbagai macam himpunan, himpunan bilangan adalah salah satu yang penting untuk dipahami.
Beberapa macam himpunan bilangan:
1 Himpunan bilangan asli = $\mathbb{N}$
2 Himpunan bilangan bulat = $\mathbb{Z}$



Gambar 7.2: Tampilan *frame* dengan *overlay pause*

Perhatikan lima tampilan halaman pdf seperti diperlihatkan pada gambar 7.2! Tampak bahwa jika halaman-halaman pdf ini muncul bergantian, kesan yang timbul adalah teks-teks di dalamnya muncul bertahap.

7.5 *Overlay Specifications* pada Daftar

Berbeda dengan *overlay pause* yang menampilkan bagian-bagian *frame* secara bertahap dari bagian awal hingga bagian akhir (berurutan), *overlay specifications* menampilkan *frame* per bagian, tetapi urutannya, yakni bagian mana yang ditampilkan di tiap halaman pdf, adalah sesuai keinginan dan pengaturan kita. Terdapat beberapa macam *overlay specifications*. Kita akan membahasnya dari yang biasa dipakai pada *list* (daftar) yang disertai penomoran ataupun yang tidak.

Untuk menambahkan *overlay specifications* pada *item-item* di dalam *environment* `enumerate` atau `itemize`, perintah `\item` diubah menjadi `\item<...>`. Titik-titik diisi sesuai urutan halaman

pdf (dari *frame* yang sama) yang ingin digunakan untuk menampilkan *item* tersebut. Cara mengisinya dijelaskan sebagai berikut:

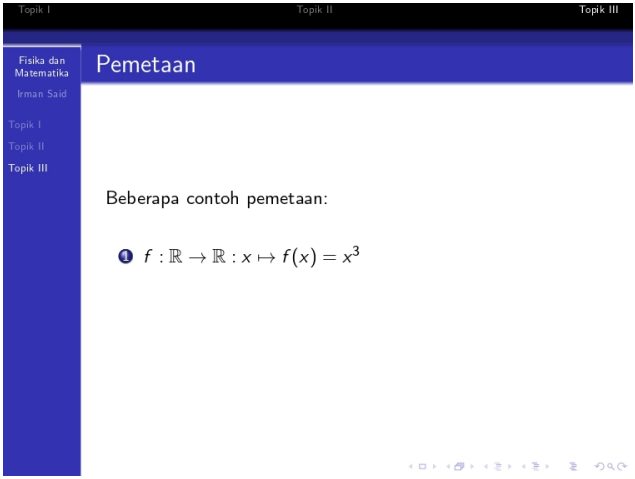
1. Jika kita menuliskan `<1-4>` maka bagian ini akan ditampilkan pada halaman pertama hingga keempat dari *frame* yang sama.
2. Jika kita menuliskan `<1,3,5>` maka bagian ini akan ditampilkan pada halaman pertama, ketiga, dan kelima dari *frame* yang sama.
3. Jika kita menuliskan `<-2,4>` maka bagian ini akan ditampilkan pada halaman pertama hingga kedua dan halaman keempat dari *frame* yang sama.
4. Jika kita menuliskan `<2,4->` maka bagian ini akan ditampilkan pada halaman kedua dan halaman keempat hingga halaman terakhir dari *frame* yang sama.
5. Jika kita menuliskan `<-2,4->` maka bagian ini akan ditampilkan pada halaman pertama hingga kedua dan halaman keempat hingga halaman terakhir dari *frame* yang sama.

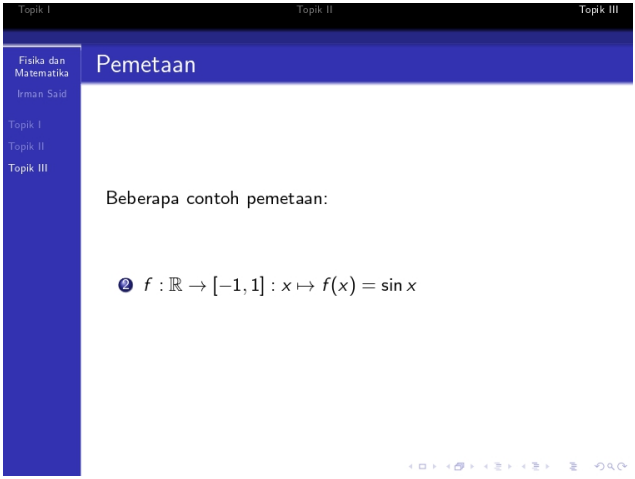
Untuk *overlay specifications* jenis ini, saat bagian atau *item* tertentu tidak ditampilkan, bagian itu hanya *tidak terlihat* tetapi *spacenya* tetap ada.

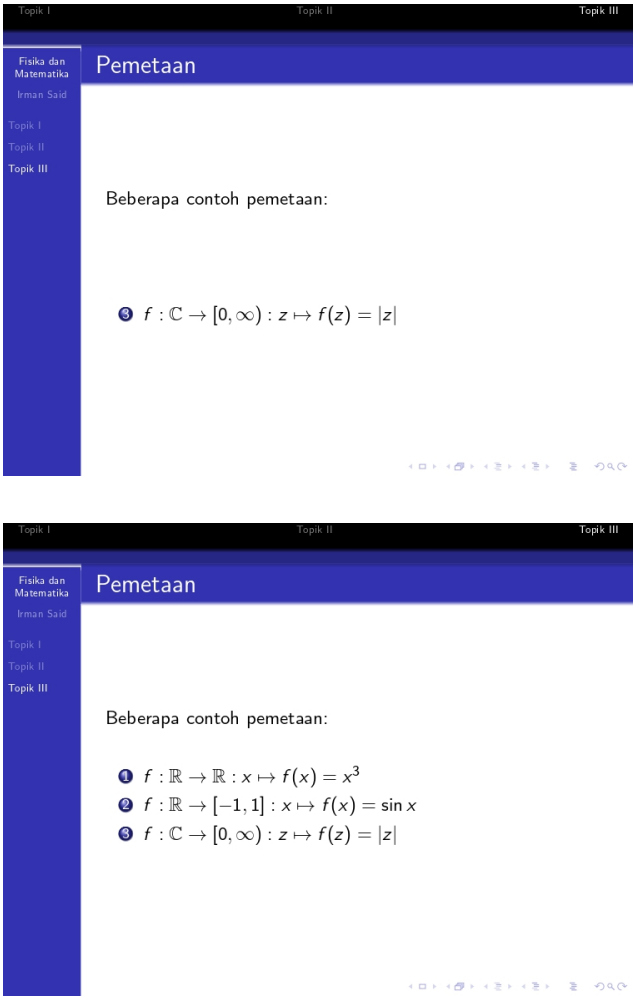
Contoh:
Dari kode yang telah dituliskan pada contoh sebelumnya, sebelum `\end{document}` tambahkan kode untuk *frame* baru berikut:

```
\begin{frame}{Pemetaan}
Beberapa contoh pemetaan:\\[0.5cm]
\begin{enumerate}
\item<1,4> $f:\mathbb{R}\to\mathbb{R}:x\mapsto f(x)=x^3$
\item<2,4> $f:\mathbb{R}\to[-1,1]:x\mapsto f(x)=\sin x$
\item<3-> $f:\mathbb{C}\to[0,\infty):z\mapsto f(z)=|z|$
\end{enumerate}
```

Hasil dari perintah untuk satu *frame* ini adalah empat halaman pdf seperti pada gambar 7.3


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Pemetaan
Beberapa contoh pemetaan:
1. $f : \mathbb{R} \to \mathbb{R} : x \mapsto f(x) = x^3$


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Pemetaan
Beberapa contoh pemetaan:
2. $f : \mathbb{R} \to [-1, 1] : x \mapsto f(x) = \sin x$

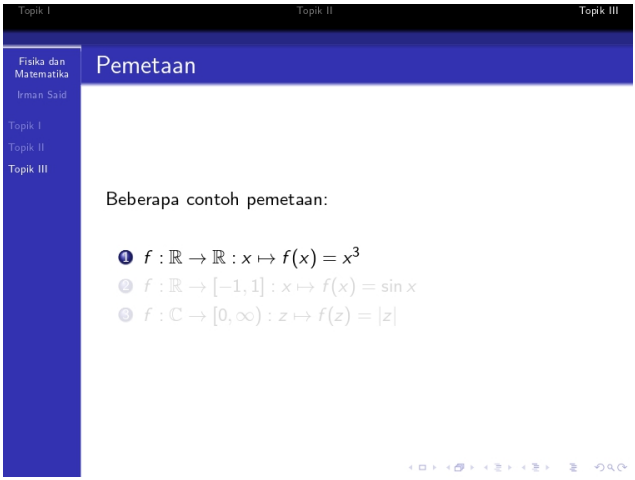


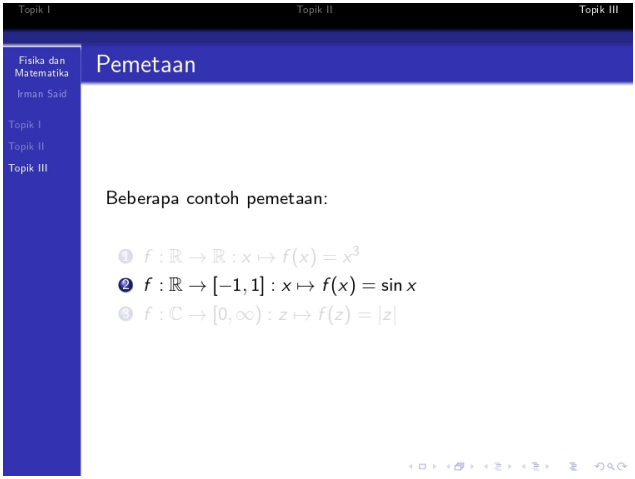
Gambar 7.3: Tampilan *frame* dengan *overlay specifications* pada daftar

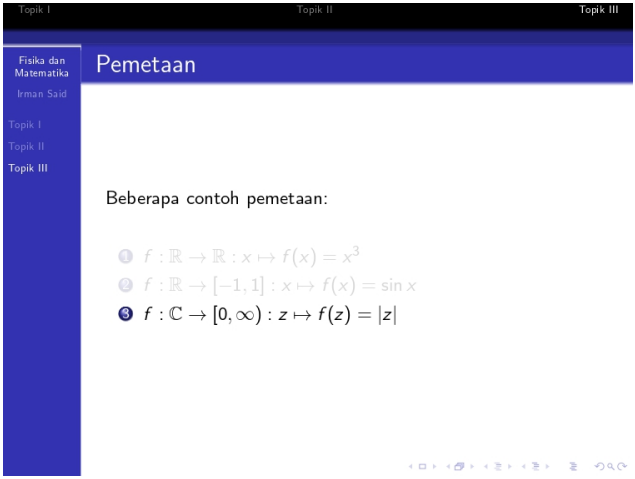
Dari contoh di atas, terlihat bahwa bagian-bagian yang tidak ditampilkan benar-benar tidak terlihat. Kita bisa membuat tidak benar-benar tidak terlihat, melainkan transparan tetapi masih terlihat. Caranya adalah pada *preamble* tambahkan perintah

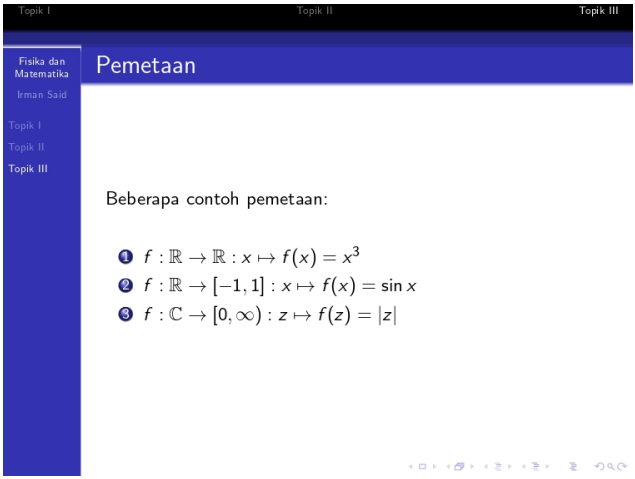
`\setbeamercovered{transparent}`!

Compile ulang dan hasil dari perintah untuk satu *frame* terakhir adalah seperti pada gambar 7.4.




Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Pemetaan
Beberapa contoh pemetaan:
1. $f : \mathbb{R} \to \mathbb{R} : x \mapsto f(x) = x^3$
2. $f : \mathbb{R} \to [-1, 1] : x \mapsto f(x) = \sin x$
3. $f : \mathbb{C} \to [0, \infty) : z \mapsto f(z) = |z|$


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Pemetaan
Beberapa contoh pemetaan:
1. $f : \mathbb{R} \to \mathbb{R} : x \mapsto f(x) = x^3$
2. $f : \mathbb{R} \to [-1, 1] : x \mapsto f(x) = \sin x$
3. $f : \mathbb{C} \to [0, \infty) : z \mapsto f(z) = |z|$



Gambar 7.4: Tampilan *frame* dengan *overlay specifications* pada daftar dengan mode transparan.

7.6 Beberapa *Overlay specifications* Lainnya

Overlay specifications selain digunakan pada daftar atau penomoran, dapat juga digunakan untuk mengatur kemunculan bagian-bagian *frame* dengan beberapa macam perintah yang berbeda. Perintah-perintah itu di antaranya adalah:

1. `\only<...>{...}`
   Titik-titik sebelah kiri diisi dengan bilangan-bilangan yang mengatur urutan kemunculan atau tampilnya bagian *frame*. Titik sebelah kanan diisi teks atau bagian yang akan ditampilkan. Penjelasan terkait pengisian titik-titik ini berlaku juga untuk perintah-perintah pada nomor-nomor berikutnya. Khusus untuk perintah `\only` ini, pada saat suatu bagian tidak dimunculkan di suatu halaman pdf, bagian itu tidak menempati ruang. Jadi bagian itu tidak hanya tidak tampak, tetapi

"benar-benar hilang".

2. `\visible<...>{...}`
   Perintah ini bekerja hampir sama dengan perintah `\only`, bedanya adalah bagian yang tidak muncul di halaman pdf tidak terlihat sepenuhnya tetapi masih menempati ruang. Jika bagian itu tidak tampak, ruang yang ditempatinya menjadi kosong tetapi tidak bisa ditempati bagian *frame* yang lain.

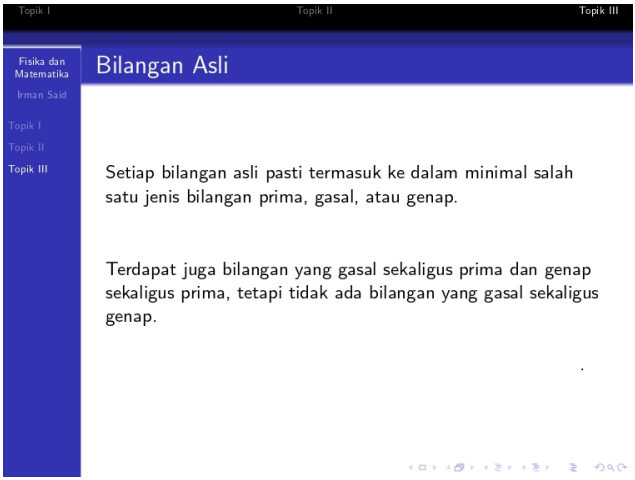
3. `\uncover<...>{...}`
   Perbedaan perintah ini dengan perintah sebelumnya adalah bahwa bagian yang tidak tampil di suatu halaman pdf masih akan menempati ruang, tetapi dengan dua kemungkinan keadaan. Keadaan pertama, bagian itu benar-benar tidak tampak jika di bagian *preamble* tidak dituliskan perintah `\setbeamercovered{transparent}`. Kedua, bagian itu masih tampak tetapi tidak jelas (transparan) jika di *preamble* dituliskan perintah `\setbeamercovered{transparent}`.

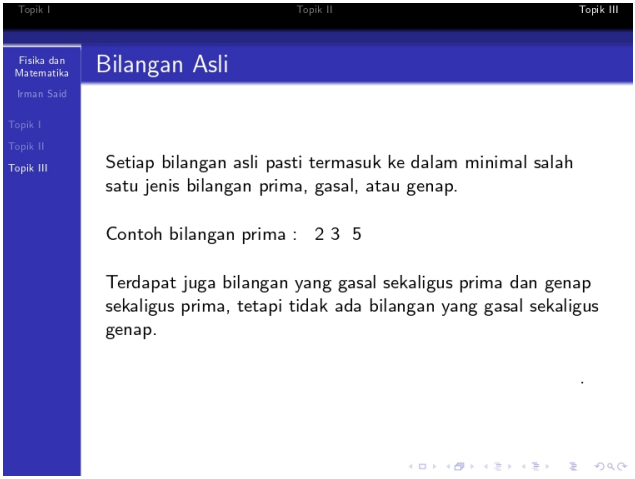
Contoh:
Hilangkan dulu perintah `\setbeamercovered{transparent}` di *preamble*. Untuk menghilangkan tidak harus menghapusnya tetapi cukup dengan menambahkan tanda `%` di sebelah kirinya. Selanjutnya dari kode yang telah dituliskan pada contoh sebelumnya, sebelum `\end{document}` tambahkan kode untuk *frame* baru berikut:

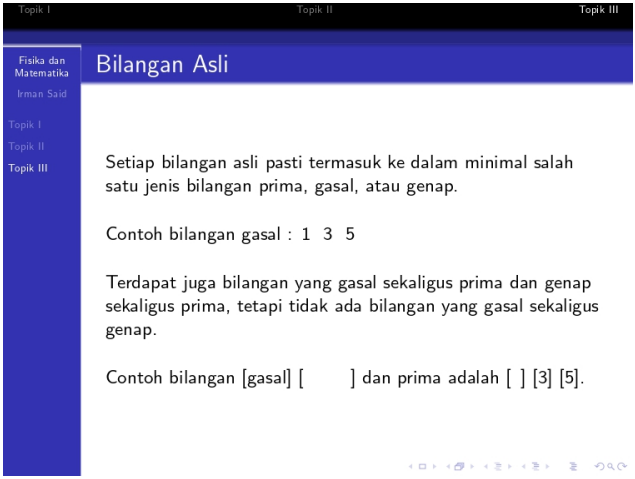
```
\begin{frame}{Bilangan Asli}
\only<1->{Setiap bilangan asli pasti termasuk ke dalam
minimal salah satu jenis bilangan prima, gasal, atau
genap.}
\\[0.5cm]
\only<2->{Contoh bilangan} \only<3>{gasal}\only<4>
{genap}\only<2>{prima} \only<2->{:}
\only<3>{1} \only<2,4>{2} \only<2,3>{3} \only<4>{4}
\only<2,3>{5} \only<4>{6}
\vspace{0.5cm}

Terdapat juga bilangan yang gasal sekaligus prima dan
```

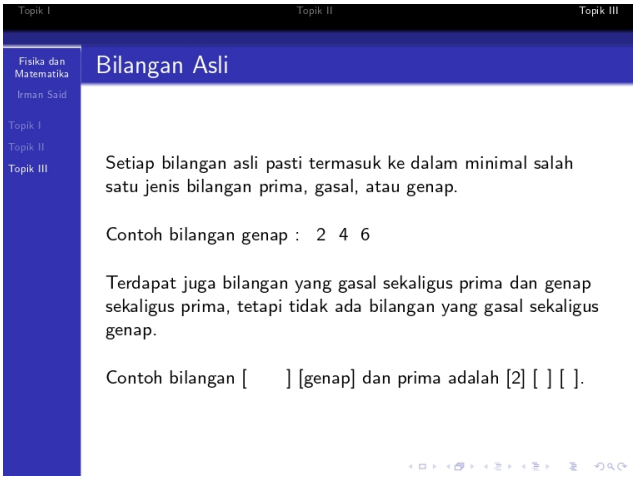
```
genap sekaligus prima, tetapi tidak ada bilangan yang
gasal sekaligus genap.
\\[0.5cm]
\visible<3->{\visible<3->{Contoh bilangan [\visible<3>
{gasal}] [\visible<4>{genap}] dan prima adalah}
[\visible<4>{2}] [\visible<3>{3}] [\visible<3>{5}]}.
\end{frame}
```

Hasil yang didapatkan dari penulisan kode untuk satu *frame* di atas adalah empat halaman pdf seperti pada gambar 7.5.




Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Bilangan Asli
Setiap bilangan asli pasti termasuk ke dalam minimal salah satu jenis bilangan prima, gasal, atau genap.
Contoh bilangan prima : 2 3 5
Terdapat juga bilangan yang gasal sekaligus prima dan genap sekaligus prima, tetapi tidak ada bilangan yang gasal sekaligus genap.

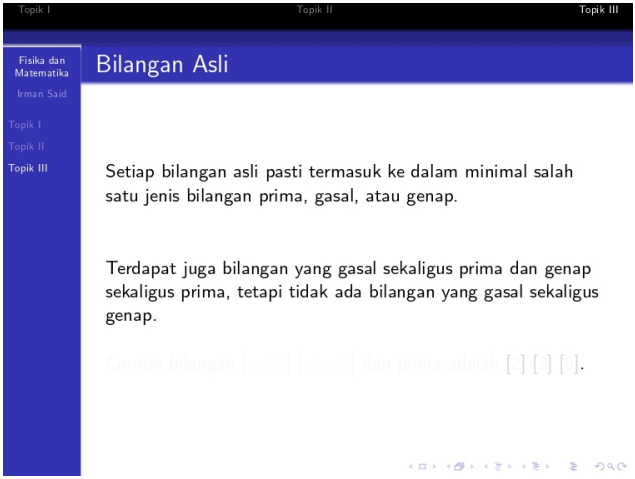

Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Bilangan Asli
Setiap bilangan asli pasti termasuk ke dalam minimal salah satu jenis bilangan prima, gasal, atau genap.
Contoh bilangan gasal : 1 3 5
Terdapat juga bilangan yang gasal sekaligus prima dan genap sekaligus prima, tetapi tidak ada bilangan yang gasal sekaligus genap.
Contoh bilangan [gasal] [ ] dan prima adalah [ ] [3] [5].

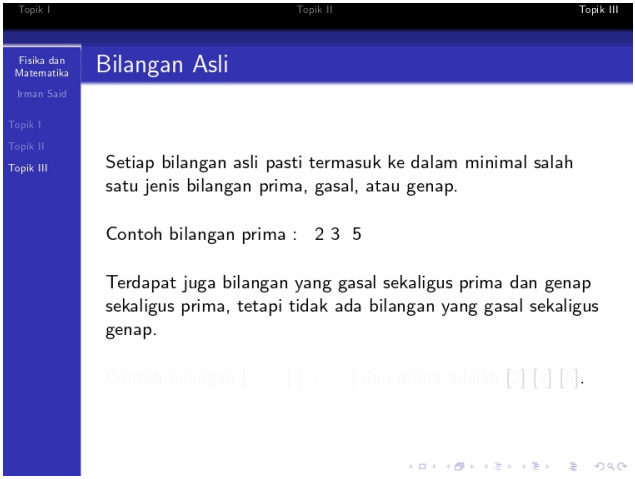


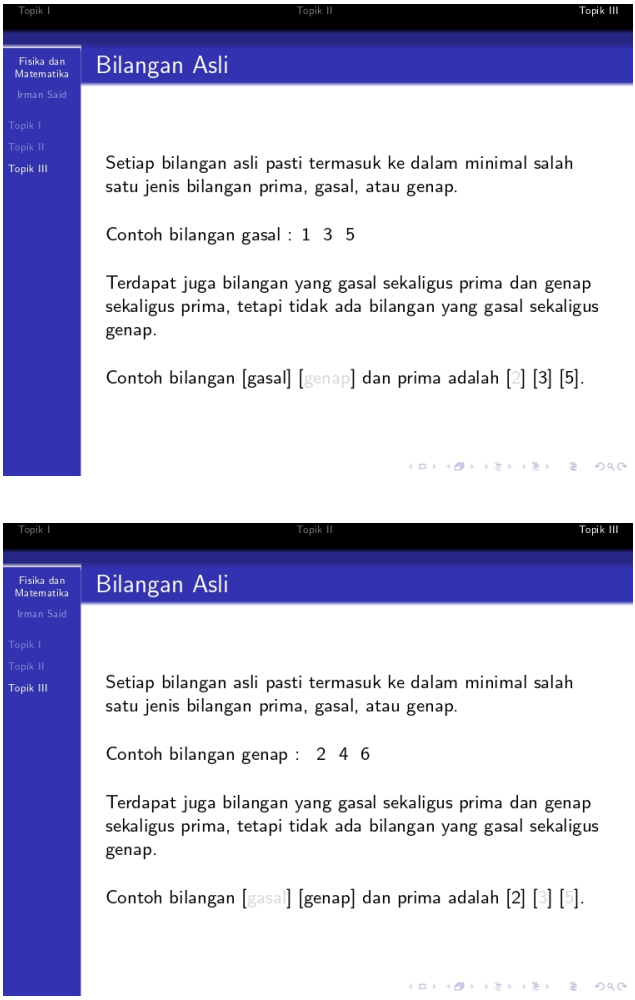
Gambar 7.5: Tampilan *frame* dengan *overlay specifications* `\only` dan `\visible`

Di atas kita telah melihat hasil *overlay spesifications* `\only` dan `\visible`. Untuk menunjukkan penggunaan *space* pada *overlay spesifications* `\visible`, pada contoh di atas kita sengaja menggunakan kurung kotak ([...]) agar terlihat jelas bahwa saat ada isinya dan saat kosong ukurannya tidak berubah. Penggunaan kurung kotak di sini hanya cara untuk menunjukkan ke pembaca, jadi secara umum kurung kotak tidak (wajib) digunakan.

Kita belum melihat penggunaan *overlay spesifications* `\uncover`. Hasil dari *overlay* ini akan sama dengan hasil dari *overlay spesifications* `\visible`, kecuali jika di *preamble* kita telah menambahkan perintah `\setbeamercovered{transparent}`. Sekarang mari kita tambahkan perintah ini di preamble! Jika sebelumnya perintah ini sudah ada dan diberi tanda `%`, hilangkan tanda `%`! Berikutnya dari kode yang telah dituliskan, ganti setiap perintah `\visible` dengan perintah `\uncover`. Hasilnya adalah seperti pada gambar 7.6


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Bilangan Asli
Setiap bilangan asli pasti termasuk ke dalam minimal salah satu jenis bilangan prima, gasal, atau genap.
Terdapat juga bilangan yang gasal sekaligus prima dan genap sekaligus prima, tetapi tidak ada bilangan yang gasal sekaligus genap.
Contoh bilangan [ ] [ ] dan prima adalah [ ] [ ] [ ].


Topik I
Topik II
Topik III
Fisika dan Matematika
Irman Said
Topik I
Topik II
Topik III
Bilangan Asli
Setiap bilangan asli pasti termasuk ke dalam minimal salah satu jenis bilangan prima, gasal, atau genap.
Contoh bilangan prima : 2 3 5
Terdapat juga bilangan yang gasal sekaligus prima dan genap sekaligus prima, tetapi tidak ada bilangan yang gasal sekaligus genap.
Contoh bilangan [ ] [ ] dan prima adalah [ ] [ ] [ ].



Gambar 7.6: Tampilan *frame* dengan *overlay specifications* `\uncover`

Bibliografi

[1] American Mathematical Society, 2002, *Users Guide for The amsmath Package (Version 2.0)*.

[2] Prastyo, I. S., 2014, *Skripsi: Kajian Teori Lie pada Osilator Harmonis dan Bandul Terbalik*, Jurusan Fisika FMIPA Universitas Gadjah Mada, Yogyakarta.

[3] Prastyo, I. S., 2017, *Tesis: Kajian Mekanika Geometrik Stokastik dan Terapannya pada Sistem-sistem Mekanis*, Jurusan Fisika FMIPA Universitas Gadjah Mada, Yogyakarta.

[4] Prathik, N., & Pahlavan, A., 2017, *Fun with Beamer: An Epic Quest To Create the Perfect Presentation*.

[5] Reckdahl, K., 2006, *Using Imported Graphics in LaTeX and pdfLaTeX*.

[6] Rosyid, M. F., 2005, *Mekanika Kuantum: Model Matematis bagi Fenomena Alam Mikroskopis, Tinjauan Non Relativistik*, Jurusan Fisika FMIPA Universitas Gadjah Mada, Yogyakarta.

[7] Rosyid, M. F., Firmansah, E., & Prabowo, Y. D., 2015, *Fisika Dasar Jilid I: Mekanika* Yogyakarta: Penerbit Periuk.

[8] Vrachimis, S., & Kyriacou, A., 2020, *Introduction to Latex - Beamer*, University of Cyprus.

[9] Walczak, Z., 2007, *Graphics in LaTeX using TikZ: TUGboat, 29(1)*, 176-179.

[10] en.wikibooks.org., 2016, 18 Juni, LaTeX, diakses pada Agustus-September 2021, dari https://en.wikibooks.org/wiki/LaTeX.

— Tentang Penulis

Irman Said Prastyo lahir pada 28 Desember 1991 di Pati, Jawa Tengah. Penulis menempuh pendidikan dasar hingga menengah di Pati kemudian melanjutkan kuliah strata satu (2010-2014) dan program master (2015-2017) di prodi fisika fakultas MIPA Universitas Gadjah Mada Yogyakarta. Penulis bekerja dan mengabdi (2017-sekarang) sebagai pengajar di jurusan fisika FST UIN Walisongo Semarang.

Motto hidup penulis :

"Jadikan dirimu penyebab orang lain bersyukur karena adanya kamu!"