

L^AT_EX for Ninjas

Prof. Walace de Almeida Rodrigues

Instituto Federal de Minas Gerais - Campus Formiga

17 de maio de 2016



Agradeço ao Prof. Diego Mello da Silva pela valiosa ajuda na elaboração do material deste curso



Sumário

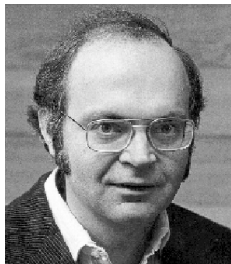
- | | | | |
|---|--------------------------|----|--------------------|
| 1 | Introdução | 7 | Matemática |
| 2 | Elementos básicos | 8 | Tabelas |
| 3 | Ambientes básicos | 9 | Algoritmos |
| 4 | Organização do documento | 10 | Gráficos |
| 5 | Bibliografia | 11 | Programação |
| 6 | Caixas | 12 | Criação de pacotes |
| | | 13 | Referências |

Introdução

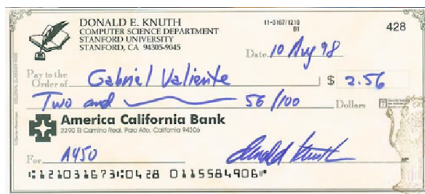
1 Introdução

- Breve história do $\text{T}_{\text{E}}\text{X}$ / \LaTeX
- Visão simplificada do $\text{T}_{\text{E}}\text{X}$
- Vantagens de usar \LaTeX
- Estrutura do curso

Apresentando: **Donald Ervin Knuth**¹



- TAOCP: *The Art of Computer Programming* (1968)
- É um dos “pais” da **Análise de Algoritmos**
- Professor Emérito da Universidade Stanford
- Vencedor do **Prêmio Turing** (1974)
- Eleito *Fellow* da *Royal Society*, de Londres (2003)
- **Criou o sistema tipográfico T_EX**



¹<http://www-cs-faculty.stanford.edu/~uno/>

- Conta-se que Knuth revisava o Volume II do TAOCP quando ficou insatisfeito com a tipografia usada pelo seu Editor.
- Ao ver o livro *Artificial Intelligence* (de Patrick Winston) produzido digitalmente, Knuth percebeu que uma composição tipográfica não era nada mais, nada menos, do que arranjar 0s e 1s (ponto **com tinta** ou **sem tinta**)
- Knuth aprendeu quais eram as regras tradicionais para tipografia matemática, o que constituía uma boa tipografia, e tanto quanto pode sobre *design*
- Todo esse trabalho resultou no **T_EX**, uma linguagem projetada para tipografia de materiais **técnicos** e **matemáticos**
- Exemplo de fórmula escrita em T_EX:

```
$$  
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}  
$$
```

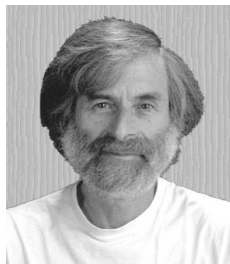
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Versões do T_EX seguem o valor de π : 3, 3.1, 3.14, 3.141, 3.141592...

1.1. Leslie Lamport

(3/4)

Apresentando: **Leslie Lamport**²

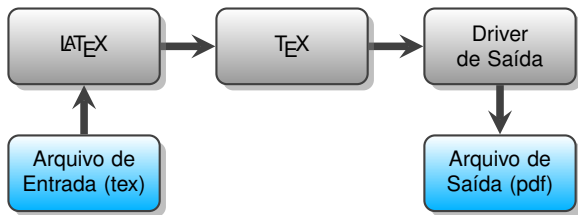


- Ph.D. em Matemática (Brandeis University, 1972), atua na computação desde 1970, Doutor Honorário em Ciência da Computação por várias universidades
- Desde 2001 trabalha na Microsoft Research
- Várias contribuições na **Teoria dos Sistemas Distribuídos**, seus *papers* estão hoje entre os mais citados no mundo
- É “pai” da **Lógica Temporal de Ações** (TPA), utilizada na descrição de comportamento de sistemas distribuídos e reativos
- Alguns notáveis: **Dijkstra Prize** (2005), **John von Neumann Medal** (2008), **Prêmio Turing** (2013), eleito **ACM Fellow** (2014)
- **Criou a linguagem de marcação L^AT_EX**

²<http://www.lamport.org/>

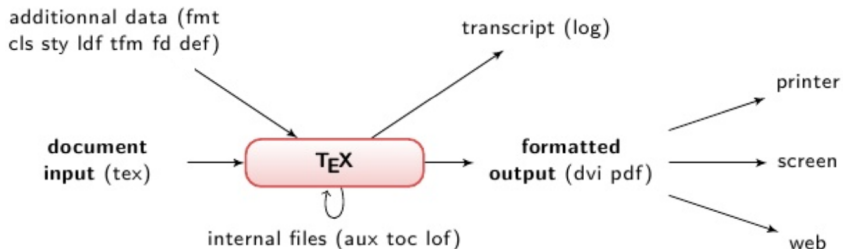
1.1. Leslie Lamport e \LaTeX

- Nos anos 70, Knuth criou o T_EX e mudou a tipografia
- Problema: T_EX não é fácil de usar
- Nos anos 80, Lamport desenvolveu uma linguagem para separar os trabalhos do *author* e do *designer*: nascia o L^AT_EX (1985)
- o L^AT_EX é um conjunto de pacotes de macros que simplificam o uso do T_EX



1.2. Esquema simplificado do T_EX

- O núcleo do nosso sistema tipográfico é o compilador de T_EX
- Outros módulos do sistema (LaTeX, BibTeX, etc) geram arquivos para o T_EX
- O próprio T_EX gera vários arquivos intermediários durante a compilação



- O *output driver* traduz a saída do T_EX para outros tipos específicos (dvi, pdf, html, postscript, etc)

1.3. Por que usar \LaTeX ao invés de \TeX

(1/2)

- \TeX foi projetada para fornecer apenas capacidades **fundamentais** de tipografia
- Operações de tipografia do \TeX são aplicadas em **nível muito baixo**, para:
 - concatenar caracteres em palavras e estas em parágrafos
 - posicionar símbolos adequadamente em fórmulas matemáticas
 - encontrar quebras de páginas adequadas automaticamente
 - lidar com notas de rodapé e outros objetos flutuantes (imagens, tabelas)
- Problema: autores preferem trabalhar em nível mais alto. Por exemplo, para centralizar algum objeto, ao invés de digitar em \TeX

```
\begingroup  
  %\rightskip=0pt plus.2\hsize  
\leftskip=\rightskip  
\parindent=0pt \parfillskip=0pt  
\noindent  
...  
\par \endgroup
```

os autores preferem simplesmente digitar em **nível mais alto** a macro do \LaTeX

```
\center ... \endcenter
```

Prós

- estilo mais consistente (*layout*, fontes, fórmulas, tabelas, etc)
- ciências exatas não são um problema (matemática, computação, química, etc)
- índices, notas de rodapé e referências são facilmente geradas
- o autor tem total domínio da estrutura do documento

Contras

- o resultado final não é imediatamente visível (★)
- curva de aprendizado
- é preciso aprender os comandos \LaTeX necessários
- a customização pode algumas vezes ser difícil

(★) \LaTeX **não** usa uma abordagem WYSIWYM (*What You See Is What You Mean*)

1.4. Planejamento

Estrutura do curso				
Módulo 0	Introdução <ul style="list-style-type: none"> Breve história do TeX / LaTeX Visão simplificada do TeX Vantagens de usar LaTeX 			
Módulo 1	Elementos básicos <ul style="list-style-type: none"> Estrutura de um documento Texto e parágrafos Tamanho Estilo Cores 	Ambientes básicos <ul style="list-style-type: none"> Definição Listas Alinhamentos Molduras Especiais <ul style="list-style-type: none"> verbatim comment 	Organização lógica <ul style="list-style-type: none"> Grupos Caixas <ul style="list-style-type: none"> figure minipage Seções, índices, anexos Referências cruzadas Rodapés 	Bibliografia <ul style="list-style-type: none"> Embarcada BibTeX JabRef
Módulo 2	Tabelas <ul style="list-style-type: none"> Idéia geral Personalizando colunas Personalizando linhas Pacote tabularx Pacote booktab Pacote array 	Matemática <ul style="list-style-type: none"> Idéia geral Símbolos e operadores Fórmulas Delimitadores Matrizes Teoremas 	Algoritmos <ul style="list-style-type: none"> Idéia geral Família de pacotes <ul style="list-style-type: none"> Pacote algorithm2e Pacote algorithm Pacote listings 	Gráficos <ul style="list-style-type: none"> Idéia geral Pacote TikZ <ul style="list-style-type: none"> Grafos Autômatos Árvores Diagramas ER Pacote smartdiagram
Módulo 3	Programação <ul style="list-style-type: none"> Idéia geral Criando comandos Criando ambientes Pacote xargs Pacote environ Pacote etools 	Criação de pacotes <ul style="list-style-type: none"> Idéia geral Estrutura de um pacote Comandos para programadores 		

Elementos básicos

2 Elementos básicos

- Estrutura de um documento
- Texto e parágrafos
- Tamanho
- Estilo
- Cores
- Grupos

2.1. Estrutura global de um documento \LaTeX

(1/6)

- Documentos em \LaTeX são arquivos no formato ASCII, portanto intercambiáveis entre vários sistemas operacionais
- Todo documento \LaTeX pode ser dividido em pelo menos duas partes:
 - 1 preâmbulo – instruções para formatação e uso de pacotes
 - 2 corpo – o conteúdo propriamente dito do documento

```
%-----  
%  Cabeçalho do documento  
%-----  
  
% Preâmbulo  
\documentclass[a4paper]{report}  
  
% Corpo  
\begin{document}  
    Oi mundo.  
\end{document}
```

2.1. Composição do preâmbulo

(2/6)

- Todo arquivo \LaTeX inicia com um preâmbulo, que contém **ao menos** o comando

```
\documentclass [opções] { classe }
```

- O parâmetro **classe** é obrigatório, e pode ser:
 - **article** , para artigos científicos
 - **report** , para relatórios técnicos e teses
 - **book** , para livros
 - **slides** , para transparências
 - **letter** , para cartas
- O parâmetro **opções** é opcional, e pode ser:
 - **a4paper** , para papel tamanho A4
 - **letterpaper** , para papel tamanho carta
 - **10pt** , para tamanho 10 pontos (padrão)
 - **11pt** , para tamanho 11 pontos
 - **12pt** , para tamanho 12 pontos
 - **twocolumn** , para texto em duas colunas
 - **twoside** , para impressão nos dois lados do papel

2.1. Composição do preâmbulo

(3/6)

- Além do comando `\documentclass`, o preâmbulo pode conter comandos

```
\usepackage[ opções ]{ pacote }
```

- O comando `\usepackage` serve para incluir pacotes \LaTeX que aumentam a capacidade de formatação. Alguns pacotes permitem opções, outros não.
- Os principais pacotes, incluindo as documentações, encontram-se no Repositório CTAN: *Comprehensive TEX Archive Network*³
- Existe uma infinidade de pacotes e veremos alguns ao longo do curso.
- Um exemplo típico de preâmbulo em \LaTeX :

```
% Linhas que começam com % são comentário em LaTeX
% Define o documento como relatório técnico em papel A4
\documentclass[a4paper]{report}

\usepackage{graphicx}      % pacote para lidar com imagens
\usepackage[brazil]{babel} % pacote para língua portuguesa
```

³<https://www.ctan.org/>

Pacote	Descrição breve
babel	gerencia regras de tipografia definidas culturalmente (russo, hebreu, português, etc)
inputenc	traduz codificações de entrada para uma “linguagem interna \LaTeX ”, permitindo acentos
fontenc	permite hifenização do texto e opera em textos contendo quaisquer caracteres na fonte
geometry	permite gerenciar facilmente o <i>layout</i> da página (tamanho das margens, etc)
graphicx	gerencia a inclusão de imagens no texto (rotação, escala, etc)
amsmath	símbolos matemáticos da <i>American Mathematical Society</i>
algorithm	ambiente para trabalhar com texto no estilo de algoritmos (atribuição, repetição, condicional, etc)
xcolor	gerencia cor de texto e de fundo

2.1. Composição do corpo

(5/6)

- Após o preâmbulo, inicia-se o documento \LaTeX propriamente dito.
- O texto do documento fica cercado entre dois comandos obrigatórios que são `\begin{document}` e `\end{document}`
- Um exemplo típico de documento em \LaTeX , com preâmbulo e corpo:

```
% Define o documento como relatório técnico em papel A4
\documentclass[a4paper]{report}

% Inclusão de pacotes
\usepackage[utf8]{inputenc}      % codificacao
\usepackage[T1]{fontenc}        % hifenizacao
\usepackage[brazil]{babel}      % tipografia da linguagem

% Corpo do documento
\begin{document}
  Isto é um texto muito, muito simples, em  $\text{\LaTeX}$ . \\
  O valor de  $\pi$  é 3.141595 \dots \\
  Fórmula de Bhaskara:  $x = \frac{-b \pm \sqrt{\Delta}}{2a}$ ,
  com  $\Delta = b^2 - 4ac$ .
\end{document}
```

■ Compilando-se o código anterior

```
\documentclass[a4paper]{report}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[brazil]{babel}
\begin{document}
  Isto é um texto muito, muito simples, em \LaTeX. \\
  O valor de  $\pi$  é 3.141595 \dots \\
  Fórmula de Bhaskara:  $x = \frac{-b \pm \sqrt{\Delta}}{2a}$ ,
  com  $\Delta = b^2 - 4ac$ .
\end{document}
```

■ Obtêm-se

Isto é um texto muito, muito simples, em \LaTeX .
O valor de π é 3.141595...
Fórmula de Bhaskara: $x = \frac{-b \pm \sqrt{\Delta}}{2a}$, com $\Delta = b^2 - 4ac$.

- Um documento \LaTeX contém apenas texto puro (ASCII), a formatação acontece através de comandos digitados ao longo desse texto
- Alguns caracteres são especiais, são tratados de forma diferenciada pelo \LaTeX , então se quiser a presença deles no texto deve gerá-los através de comandos da seguinte maneira:

Caractere especial	Texto \LaTeX
\$	<code>\\$</code>
%	<code>\%</code>
&	<code>\&</code>
#	<code>\#</code>
{	<code>\{</code>
}	<code>\}</code>
/	<code>\slash</code>
\	<code>\textbackslash</code>

- Se o pacote **fontenc** não tiver sido declarado no preâmbulo, a acentuação pelo L^AT_EX também deverá ser gerada através de comandos da seguinte maneira:

Caractere acentuado	Texto L ^A T _E X
ç	<code>\c{c}</code>
á	<code>\'{a}</code>
à	<code>\`{a}</code>
â	<code>\^{a}</code>
ã	<code>\~{a}</code>
ü	<code>\" {u}</code>

- Evite amolações incluindo no preâmbulo dois pacotes que tratam acentuação:

```
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
```

- O pacote **inputenc** é orientado para a entrada, permite a entrada de caracteres acentuados diretamente do teclado, traduzindo-os para o L^AT_EX
- O pacote **fontenc** é orientado para a saída, gerando os caracteres adequados para a impressão, de acordo com a fonte escolhida, e controla hifenizações

- A quantidade de espaços digitados entre as palavras não faz diferença num documento \LaTeX , sempre será contado apenas um espaço em branco.
- Todo texto que não esteja separado por uma linha em branco ou por comandos que separam parágrafos forma um único parágrafo.
- A quantidade de linhas em branco entre os parágrafos não faz diferença num documento \LaTeX , sempre será contada apenas uma linha em branco.

```
\begin{document}
Exemplo  bizarro      de
      texto      muito
mal
                                formatado .

Linha branca , muda
parágrafo .
\end{document}
```

Exemplo bizarro de texto muito
mal formatado.

Linha branca, muda parágrafo.

- O tamanho padrão para espaçamento entre parágrafos é de **apenas 1pt**.
- Para configurar o tamanho do espaçamento entre parágrafos, inclua no preâmbulo o comando:

```
\setlength\parskip{tamanho}
```

especificando o tamanho com uma das unidades: pt, mm, cm, pol, ...

- O comando `\par` oferece outro modo para forçar a mudança de parágrafo.

```
\begin{document}
Exemplo  bizarro      de
      texto  muito  \par
mal
                                formatado .
```

Linha branca, muda
parágrafo.

```
\end{document}
```

Exemplo bizarro de texto muito
mal formatado.

Linha branca, muda parágrafo.

- Normalmente todos parágrafos iniciam com indentação. O espaçamento padrão para a indentação é de 15pt.
- Para configurar o espaçamento de indentação, inclua no preâmbulo o comando:

```
\setlength\parindent{tamanho}
```

especificando o tamanho com uma das unidades: pt, mm, cm, pol, ...

- Dentro de estruturas do documento (capítulos, seções, etc), o primeiro parágrafo inicia sem indentação. Esse padrão pode ser modificado pela inclusão do pacote **indentfirst** no preâmbulo.
- O comando **\indent** adiciona espaçamento horizontal do mesmo tamanho da indentação normal
- o comando **\noindent** retira a indentação do local onde ela deveria aparecer.

Existem diversas formas para inserir espaçamentos ou quebras no texto:

Comando	Resultado
<code>\\</code>	Inserir quebra de linha
<code>\\[tamanho]</code>	Inserir quebra de linha e espaço vertical
<code>\newline</code>	Inserir quebra de linha
<code>\linebreak</code>	Inserir quebra de linha e ajusta a linha anterior
<code>\newpage</code>	Inserir quebra de página
<code>\smallskip</code>	Espaço pequeno entre parágrafos
<code>\medskip</code>	Espaço médio entre parágrafos
<code>\bigskip</code>	Espaço grande entre parágrafos
<code>\vspace{tamanho}</code>	Espaço vertical
<code>\hspace{tamanho}</code>	Espaço horizontal
<code>\hfill</code>	Preenche a linha com espaços em branco
<code>\dotfill</code>	Preenche a linha com pontos
<code>\<espaço></code>	Inserir um espaço em branco manualmente

O tamanho é especificado por um inteiro seguindo da unidade: pt, mm, cm, pol, ...

2.3. Tamanho

Dez ambientes estão disponíveis para mudar o tamanho da fonte:

Comando	Tamanho
<code>\begin{tiny} exemplo \end{tiny}</code>	exemplo
<code>\begin{scriptsize} exemplo \end{scriptsize}</code>	exemplo
<code>\begin{footnotesize} exemplo \end{footnotesize}</code>	exemplo
<code>\begin{small} exemplo \end{small}</code>	exemplo
<code>\begin{normalsize} exemplo \end{normalsize}</code>	exemplo
<code>\begin{large} exemplo \end{large}</code>	exemplo
<code>\begin{Large} exemplo \end{Large}</code>	exemplo
<code>\begin{LARGE} exemplo \end{LARGE}</code>	exemplo
<code>\begin{huge} exemplo \end{huge}</code>	exemplo
<code>\begin{Huge} exemplo \end{Huge}</code>	exemplo

Também é possível usar comandos do tipo `\tiny {exemplo}` ou `{\tiny exemplo}`

Existem muitas opções e o número pode aumentar com a inclusão de pacotes:

Comando	Estilo	Nome
<code>\underline</code> {exemplo}	<u>exemplo</u>	sublinhado
<code>\emph</code> {exemplo}	<u>exemplo</u>	ênfático
<code>\textit</code> {exemplo}	<i>exemplo</i>	itálico
<code>\textsl</code> {exemplo}	<i>exemplo</i>	inclinado
<code>\textbf</code> {exemplo}	exemplo	negrito
<code>\texttt</code> {exemplo}	exemplo	máquina de escrever
<code>\textsc</code> {exemplo}	EXEMPLO	caixa alta
<code>\textrm</code> {exemplo}	exemplo	romano
<code>\textsf</code> {exemplo}	exemplo	serifado

Neste curso empregamos `\usepackage{ulem}` para outras versões de sublinhado. Este pacote também modifica o ênfático para sublinhado, como no exemplo. Normalmente o ênfático tem aparência de itálico e se for essa a escolha inclua a opção normal em na inclusão do pacote:

```
\usepackage [normal] { ulem }
```

As opções de estilo implementadas pelo pacote `ulem` são:

Comando	Estilo	Nome
<code>\uline{exemplo}</code>	<u>exemplo</u>	sublinhado
<code>\uuline{exemplo}</code>	<u><u>exemplo</u></u>	sublinhado duplo
<code>\uwave{exemplo}</code>	<u>exemplo</u>	sublinhado ondulado
<code>\sout{exemplo}</code>	exemplo	riscado
<code>\xout{exemplo}</code>	exemplo	muito riscado
<code>\dashuline{exemplo}</code>	<u>exemplo</u>	sublinhado traçado
<code>\dotuline{exemplo}</code>	<u>exemplo</u>	sublinhado pontilhado

Geralmente os estilos podem ser combinados, então cabe ao usuário escolher combinações que façam sentido. Exemplo:

```
\textbf{\textit{negrito itálico}}
```

Alternativamente, os estilos comuns também podem ser utilizados na forma de “comando de estado”:

Comando	Estilo	Nome
<code>{\em exemplo}</code>	<u>exemplo</u>	ênfatisado
<code>{\it exemplo}</code>	<i>exemplo</i>	itálico
<code>{\sl exemplo}</code>	<i>exemplo</i>	inclinado
<code>{\bf exemplo}</code>	exemplo	negrito
<code>{\tt exemplo}</code>	exemplo	máquina de escrever
<code>{\sc exemplo}</code>	EXEMPLO	caixa alta
<code>{\rm exemplo}</code>	exemplo	romano
<code>{\sf exemplo}</code>	exemplo	serifado

- O pacote **color** oferece o essencial e define as 8 cores básicas:

black		white		red		green	
blue		cyan		magenta		yellow	

- Para tratar cores, entretanto, é melhor incluir o pacote **xcolor** que estende o anterior com novas cores e acrescenta comandos para colorir tabelas:

```
\usepackage[usenames, table]{xcolor}
```

A opção **usenames** define nomes para as cores padrão

A opção **table** oferece recursos para colorir tabelas⁴

⁴Ver Seção 8.

- Cor do texto: `\textcolor{cor}{texto}` ou `\color{cor}{texto}`

Este `\textcolor{blue}{exemplo}` mostra `\textcolor{red}{como}` usar cores no `\LaTeX` `\color{cyan}` por meio de `\color{red}` comandos do pacote `\textbf{\texttt{xcolor}}`

Este exemplo mostra como usar cores no \LaTeX por meio de comandos do pacote `xcolor`

- Cor do fundo: `\colorbox{cor}{texto}`

`\colorbox{yellow}{Exemplo}` super
`\colorbox{red}{\textcolor{white}{simples}}`
 de como `\colorbox{black}{\color{cyan}podemos}` usar cores no
`\colorbox{blue}{\textcolor{green}{\LaTeX}}`

Exemplo super simples de como podemos usar cores no \LaTeX .

- Definindo cores: `\definecolor{nome}{modelo}{especificação}`

- Modelos: $\left\{ \begin{array}{l} \text{Gray (Escala de Cinza)} \\ \text{RGB (Red, Green, Blue)} \\ \text{CMYK (Cyan, Magenta, Yellow, Black)} \end{array} \right.$

```
\definecolor{MeuCinza}{gray}{0.30}           % 1 escala
\definecolor{VerdeEscuro}{rgb}{0, 0.33, 0}    % 3 escalas
\definecolor{Laranja}{cmyk}{0,0.5,1,0}        % 4 escalas

\textcolor{VerdeEscuro}{Verde escuro por RGB}
\colorbox{MeuCinza}{\textcolor{white}{Cinza por Gray}}
\textcolor{Laranja}{Laranja por CMYK}
```

Verde escuro por RGB

Cinza por Gray

Laranja por CMYK

- Consulte paletas na web para auxiliar na criação de cores ⁵

⁵latexcolor.com ou <http://bcb.med.usherbrooke.ca/images/palette.png> 🔍 ↻

- Também é possível especificar novas cores diretamente, usando um dos 3 modelos vistos ou através da mistura de cores já definidas

Forma: `\color{nome-da-cor!porcentagem-na-mistura}`

```
{\color[rgb]{0.5,0,0.5} == purple} \\
... agora misturando: \\
{\color{blue!40} == blue!40} \\
{\color{red!60!yellow} == red!60 + yellow!100} \\
{\color{blue!60!black!40!green} == blue!60 + black!40 + green!100}
```

`== purple`

`... agora misturando:`

`== blue!40`

`== red!60 + yellow!100`

`== blue!60 + black!40 + green!100`

- Comando para batizar cores criadas por mistura

Forma: `\colorlet{nome-da-cor}{mistura}`

2.6. Grupos

- **Grupo** é uma massa de texto que inicia com `{` e termina com `}` ⁶
- O conceito de **grupo** é típico do $\text{T}_{\text{E}}\text{X}$ e simplifica o trabalho do compilador. O grupo delimita uma região do texto como campo de ação para certos comandos. Mudanças de estado causadas dentro de um grupo são descartadas quando o grupo termina.

Exemplo:

```
{grupo com \bf exemplo de mudança}
```

o comando de estado `\bf`, que ativa o negrito, só vai atuar até o final do grupo, porque depois a mudança de estado será descartada.

- Entender o conceito de grupo será importante depois para a utilização de recursos mais avançados do $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$

⁶Também pode iniciar com `\bgroup` e terminar com `\egroup`

Ambientes básicos

3 Ambientes básicos

- Definição
- Listas
- Alinhamentos
- Molduras
- Especiais

3.1. Ambientes: que são e para que?

- Um ambiente \LaTeX é uma região do texto delimitada pelos comandos:

```
\begin{ambiente}  
...  
\end{ambiente}
```

- Os ambientes são utilizados pelo \LaTeX para prover diversos tipos de recursos de formatação, além de permitir ao autor definir escopos de ação para comandos específicos
- Alguns recursos úteis providos com o uso de ambientes:
 - listas: **itemize**, **enumerate**, **description**
 - alinhamentos: **center**, **flushleft**, **flushright**
 - desligar formatação: **verbatim**
 - marcar comentários: **comment**
 - molduras: **framed**
- Existem diversos outros tipos de ambientes além desses. Alguns ambientes importantes serão abordados no decorrer deste curso.

■ Exemplo simples de lista:

Os objetivos do trabalho são:

```
\begin{itemize}
  \item Ensinar os princípios de \LaTeX\ aos alunos do IFMG
  \item Melhorar a qualidade da editoração dos trabalhos
  \item Disseminar o uso desta ferramenta
\end{itemize}
```

Os objetivos do trabalho são:

- Ensinar os princípios de \LaTeX aos alunos do IFMG
- Melhorar a qualidade da editoração dos trabalhos
- Disseminar o uso desta ferramenta

- É comum o aninhamento de listas:

Para iniciar o sistema:

```
\begin{itemize}
  \item Energize o computador e pressione o botão Ligar
  \item No gerenciador de \textit{boot}, escolha:
    \begin{itemize}
      \item Linux, se for \textit{dual boot};
      \item Windows, se for único sistema disponível.
    \end{itemize}
\end{itemize}
```

Para iniciar o sistema:

- Energize o computador e pressione o botão Ligar
- No gerenciador de *boot*, escolha:
 - Linux, se for *dual boot*;
 - Windows, se for único sistema disponível.

- É simples alterar o símbolo de cada item:

% Exemplos diversos

Enumerando texto estranho:

```
\begin{itemize}
  \item[$\bullet$] bolinha
  \item[$\pi$] pi
  \item[$\Delta$] delta
  \begin{itemize}
    \item[\textbf{5.1}]
      texto
    \item[\textbf{5.2}]
      texto
  \end{itemize}
  \item[\textcircled{a}]
    círculo
  \item[\textcircled{b}]
    círculo
  \item[\textcircled{c}]
    círculo
\end{itemize}
```

Enumerando texto estranho:

- bolinha

π pi

Δ delta

5.1 texto

5.2 texto

Ⓐ círculo

Ⓑ círculo

Ⓒ círculo

■ Exemplo simples de lista:

Os objetivos do trabalho são:

```
\begin{enumerate}
  \item Ensinar os princípios de \LaTeX\ aos alunos do IFMG
  \item Melhorar a qualidade da editoração dos trabalhos
  \item Disseminar o uso desta ferramenta
\end{enumerate}
```

Os objetivos do trabalho são:

- 1 Ensinar os princípios de \LaTeX aos alunos do IFMG
- 2 Melhorar a qualidade da editoração dos trabalhos
- 3 Disseminar o uso desta ferramenta

- É simples alterar o sistema da enumeração:

Os objetivos do trabalho são:

```
\begin{enumerate}[(a)]  
  \item Ensinar os princípios de \LaTeX\ aos alunos do IFMG  
  \item Melhorar a qualidade da editoração dos trabalhos  
  \item Disseminar o uso desta ferramenta  
\end{enumerate}
```

Os objetivos do trabalho são:

- (a) Ensinar os princípios de \LaTeX aos alunos do IFMG
- (b) Melhorar a qualidade da editoração dos trabalhos
- (c) Disseminar o uso desta ferramenta

- É simples iniciar a enumeração partindo de um número arbitrário

Os objetivos do trabalho são:

```
\begin{enumerate} \setcounter{enumi}{5}  
  \item Ensinar os princípios de \LaTeX\ aos alunos do IFMG  
  \item Melhorar a qualidade da editoração dos trabalhos  
  \vspace{0.5cm}  
  \item Disseminar o uso desta ferramenta  
\end{enumerate}
```

Os objetivos do trabalho são:

- 6 Ensinar os princípios de \LaTeX aos alunos do IFMG
- 7 Melhorar a qualidade da editoração dos trabalhos
- 8 Disseminar o uso desta ferramenta

■ Exemplo simples de lista:

Os objetivos do trabalho são:

```
\begin{description}
  \item[Ensinar] os princípios de \LaTeX\ aos alunos do IFMG
  \item[Melhorar] a qualidade da editoração dos trabalhos
  \item[Disseminar] o uso desta ferramenta
\end{description}
```

Os objetivos do trabalho são:

Ensinar os princípios de \LaTeX aos alunos do IFMG

Melhorar a qualidade da editoração dos trabalhos

Disseminar o uso desta ferramenta

3.2. Pacotes úteis para personalizar listas

(8/8)

Pacote	Provê	Funcionalidade
paralist	ambientes compactitem compactenum	Gerar lista compacta
	ambientes inparaitem inparaenum	Gerar lista em linha
	ambientes asparaitem asparaenum	Gerar lista de parágrafos
	enumitem	Personalizar símbolo, espaçamento, etc

3.3. Alinhamentos

■ Centralizado

```
\begin{center}  
  Texto centralizado.  
\end{center}
```

Texto centralizado.

■ Esquerda

```
\begin{flushleft}  
  Texto à esquerda.  
\end{flushleft}
```

Texto à esquerda.

■ Direita

```
\begin{flushright}  
  Texto à direita.  
\end{flushright}
```

Texto à direita.

3.4. Molduras

- O ambiente **framed** desenha um *frame* ao redor do texto destacado. Para usar é preciso incluir `\usepackage{framed}` no preâmbulo.

O ambiente `\textbf{framed}` insere moldura ao redor de um texto

```
\begin{framed}
  Regra  $\frac{3}{8}$  de Simpson:
  
$$I_3 = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3)$$

\end{framed}
```

Para colocar `\textbf{frame}` em algumas palavras use `\texttt{\textbackslash fbox\{\dots\}}`. `\fbox{Exemplo}` de como `\fbox{colocar}` texto em `\textit{frames}`.

O ambiente **framed** insere moldura ao redor de um texto.

Regra 3/8 de Simpson: $I_3 = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3)$

Para colocar **frame** em algumas palavras use `\fbox{...}`.

3.5. Desligar formatação

(1/2)

- O ambiente **verbatim** é útil para escrever textos não compilados

```
\begin{verbatim}
```

A fórmula de Bháskara é $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$,

codificada em **LaTeX**.

```
\end{verbatim}
```

A fórmula de Bháskara é $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$,

codificada em **LaTeX**.

- Observe que o texto escrito dentro do ambiente **verbatim** não foi compilado pelo **LaTeX** e apareceu literalmente no documento final
- Existe uma alternativa em linha implementada como macro **TeX** que causa o mesmo efeito do ambiente **verbatim** e pode ser usada na forma **\verb|...|**

- O ambiente `comment` é útil na fase de escrita do documento e testes de compilação para marcar regiões do documento como comentário. Para usar é preciso incluir `\usepackage{comment}` no preâmbulo.

```
\begin{comment}
Gera: \[2mm]
\begin{moldura}[white][10.9cm][black]
  \begin{figure}
    \includegraphics[scale=0.40]{fig/foto1.png}
  \end{figure}
\end{moldura}
\smallskip
\end{comment}
```

- O texto dentro do ambiente `comment` é tratado como comentário pelo \LaTeX e não gera nenhum resultado no documento final.

Organização do documento

4 Organização do documento

- Layout das páginas
- Seções e índices
- Notas de rodapé
- Referências cruzadas

4.1. Como o \LaTeX formata as páginas

(1/2)

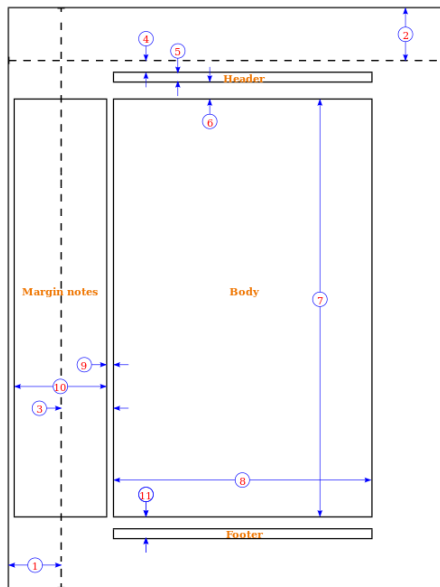
- O *layout* da página em \LaTeX é definido por vários parâmetros internos, cada um correspondendo ao tamanho de um elemento da página medido em pontos (pt)
- Esse *layout* é, evidentemente, afetado diretamente pelo tipo de documento definido em `\documentclass`
- Normalmente recomenda-se **não** modificar o padrão estabelecido mas, caso seja necessário, existem pacotes que auxiliam nessa tarefa:

Pacote	Descrição breve
layout	define o comando <code>\layout</code> que apresenta um sumário dos elementos que definem o formato da página
showframe	desenha um diagrama simples que mostra o <i>layout</i> definido
geometry	permite alterar facilmente os elementos que definem o <i>layout</i>

- Os pacotes `layout` e `showframe` são úteis para *debug* e o pacote `geometry` auxilia na configuração do *layout*

4.1. Como o \LaTeX formata as páginas

(2/2)



1. $72\text{pt} + \text{\hoffset}$
 2. $72\text{pt} + \text{\voffset}$
 3. $\text{\oddsidemargin} = 31\text{pt}$
 4. $\text{\topmargin} = 20\text{pt}$
 5. $\text{\headheight} = 12\text{pt}$
 6. $\text{\headsep} = 25\text{pt}$
 7. $\text{\textheight} = 592\text{pt}$
 8. $\text{\textwidth} = 390\text{pt}$
 9. $\text{\marginparsep} = 10\text{pt}$
 10. $\text{\marginparwidth} = 35\text{pt}$
 11. $\text{\footskip} = 30\text{pt}$
- $\text{\marginparpush} = 7\text{pt}$ (não mostrado)
 - $\text{\hoffset} = 0\text{pt}$
 - $\text{\voffset} = 0\text{pt}$
 - $\text{\paperwidth} = 597\text{pt}$
 - $\text{\paperheight} = 845\text{pt}$

A forma do documento é fixada no preâmbulo:

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

%----- mandatórios
\usepackage[brazil]{babel}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}

%----- outros
\usepackage{times}      % fontes
\usepackage{hyperref}  % links
\usepackage{amsmath,amssymb,amsfonts} % matematica

%----- o conteúdo começa aqui:
\title{Algumas explicações sobre o \LaTeX}
\author{Wallace Rodrigues}
\date{\today} % mais claro impossível
```

Uma vez definida a forma, o autor pode se concentrar no conteúdo do documento:

```
\begin{document}
\maketitle

\begin{abstract}
  O objetivo deste documento é ajudar você a entender algumas
  coisas.
\end{abstract}

\section{Introdução}
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Maecenas pretium urna ut nisl semper sed mattis erat interdum.
Vestibulum eget massa nisi. Donec feugiat consequat leo,
a vehicula est imperdiet at.

\section{O que você pode fazer?}
Quase tudo, por exemplo, se você precisar é muito fácil trabalhar
no modo matemático. É claro que  $\forall x \neq 0, \frac{x^2}{x} = x$ .
Mas isso fica mais evidente quando está bem escrito.
\end{document}
```

A estrutura lógica do documento é determinada nas seções utilizadas pelo autor:

Algumas explicações sobre o \LaTeX

Wallace Rodrigues
27 de janeiro de 2016

Resumo

O objetivo deste documento é ajudar você a entender algumas coisas.

1 Introdução

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas pretium urna ut nisl semper sed mattis erat interdum. Vestibulum eget massa nisi. Donec feugiat consequat leo, a vehicula est imperdiet at.

2 O que você pode fazer?

Quase tudo, por exemplo, se você precisar é muito fácil trabalhar utilizando o modo matemático. É claro que $\forall x \neq 0, \frac{x^2}{x} = x$. Mas isso fica ainda mais evidente quando vem bem escrito no documento.

- Para criar seções específicas no documento, usar:

Comando	Nível
<code>\chapter{...}</code>	0
<code>\section{...}</code>	1
<code>\subsection{...}</code>	2
<code>\subsubsection{...}</code>	3

Nota: `\chapter{...}` existe somente para as classes `book` e `report`

- Para incluir índices para seções, figuras e tabelas, usar:

Comando	Descrição
<code>\tableofcontents{...}</code>	gera o índice com as seções
<code>\listoffigures{...}</code>	gera o índice de figuras
<code>\listoftables{...}</code>	gera o índice de tabelas

Nota: outros tipos de índices podem ser gerados por pacotes específicos

4.3. Notas de rodapé

- Acrescentar notas de rodapé é fácil: use o comando `\footnote{...}`

Criar notas de rodapé é fácil `\footnote`{Um exemplo de nota.}.

- Para numerar as notas de rodapé por seção, acrescente ao preâmbulo:

```
\makeatletter  
\@addtoreset{footnote}{section}  
\makeatother
```

- Para numerar as notas de rodapé por página, acrescente ao preâmbulo:

```
\usepackage{perpage}  
\MakePerPage{footnote}
```

4.4. Referências cruzadas

- Quase tudo numerado (seções, tabelas, figuras, etc) pode ser referenciado:

Comando	Descrição
<code>\label{marca}</code>	coloca uma marca no objeto
<code>\ref{marca}</code>	referencia o objeto marcado
<code>\pageref{marca}</code>	imprime o número da página onde está o objeto marcado

- Adicione um prefixo no nome da marca para indicar o tipo do objeto marcado:

Prefixo	Objeto
sec:	seção
fig:	figura
tab:	tabela
itm:	item de lista

Prefixo	Objeto
eq:	equação
lst:	código
alg:	algoritmo
app:	apêndice

- Exemplo:

Veja a Figura~`\ref{fig:exemplo}` na página~`\pageref{fig:exemplo}`.

Bibliografia

5 Bibliografia

- Idéia geral
- Bibliografia embarcada
- BibTeX

5.1. Citações e Referências Bibliográficas

- Acrescentar referências ao documento é uma parte importante em trabalhos acadêmicos e felizmente o \LaTeX simplifica muito essa tarefa
- Basicamente, o autor cria uma *tag* para cada obra consultada, fornece as informações necessárias sobre cada obra, e finalmente referencia as obras armazenadas recorrendo às *tags*.
- Existem duas formas básicas para lidar com as informações sobre as obras:
 - Ambiente **thebibliography**, as informações ficam embutidas no arquivo **.tex** do documento
 - Módulo **BibTeX**, as informações ficam armazenadas num arquivo texto auxiliar **.bib** que depois será consultado pelo módulo para gerar os dados necessários para o \LaTeX
- Em ambos os casos é possível fazer citações para as obras consultadas usando o comando **\cite**

- O ambiente **thebibliography** é simples de usar, mas exige que o usuário escreva os dados da obra por conta própria. As obras são inseridas dentro do ambiente através do comando **\bibitem**

```
\section{Introdução}
Segundo \cite{barroso}, os sistemas lineares são classificados em possíveis
determinados,
possíveis indeterminados e impossíveis.

\subsection{Resolução por métodos diretos}
Sistemas lineares podem ser resolvidos por métodos diretos e iterativos. \cite{
ruggiero}
citam como métodos diretos a Decomposição LU e a eliminação de Gauss.

\subsection{Resolução por métodos iterativos}
\cite{barroso} discutem em sua obra dois dos métodos iterativos mais conhecidos
para resolução de sistemas lineares: o método de Gauss-Seidel e o método de Jacobi.

\begin{thebibliography}{1}
  \bibitem[Barroso e Campos]{barroso}
    BARROSO, Leonidas; CAMPOS FILHO, Frederico Ferreira.
    \newblock Cálculo Numérico (Com Aplicações). 2ª edição.
    \newblock Editora Harbra, 1987. ISBN: 85-29400-89-5

    \bibitem[Ruggiero e Lopes]{ruggiero}
    RUGGIERO, Márcia; LOPES, Vera Lúcia da Rocha.
    \newblock Cálculo Numérico – Aspectos Teóricos e Computacionais. 2ª edição.
    \newblock Editora Pearson Makron, 1996. ISBN: 978-85-346-0204-4.
\end{thebibliography}
```

O resultado final no documento é:

1 Introdução

Segundo [Barroso e Campos], os sistemas lineares são classificados em possíveis determinados, possíveis indeterminados e impossíveis.

1.1 Resolução por métodos diretos

Sistemas lineares podem ser resolvidos por métodos diretos e iterativos. [Ruggiero e Lopes] citam como métodos diretos a Decomposição LU e a eliminação de Gauss.

1.2 Resolução por métodos iterativos

[Barroso e Campos] discutem em sua obra dois dos métodos iterativos mais conhecidos para resolução de sistemas lineares: o método de Gauss-Seidel e o método de Jacobi.

Referências

[Barroso e Campos] BARROSO, Leonidas; CAMPOS FILHO, Frederico Ferreira. Cálculo Numérico (Com Aplicações). 2ª edição. Editora Harbra, 1987. ISBN: 85-29400-89-5

[Ruggiero e Lopes] RUGGIERO, Márcia; LOPES, Vera Lúcia da Rocha. Cálculo Numérico - Aspectos Teóricos e Computacionais. 2ª edição. Editora Pearson Makron, 1996. ISBN: 978-85-346-0204-4.

- O módulo **BibTeX** oferece comodidade para tratar longas listas de referências
- As informações sobre as obras ficam armazenadas num arquivo texto **.bib** que funciona como base de dados para o módulo **BibTeX**
- A estrutura do arquivo **.bib** é muito simples e ele pode ser facilmente editado. Exemplo de entrada nesse arquivo:

```
@article{greenwade93,  
  author = "George D. Greenwade",  
  title = "The Comprehensive Tex Archive Network ({CTAN})",  
  year = "1993",  
  journal = "TUGBoat",  
  volume = "14",  
  number = "3",  
  pages = "342--351"  
}
```

- Cada entrada no arquivo `.bib` começa com a declaração do tipo da referência, na forma `@tipo`.
- O **BibTeX** oferece uma variedade enorme de tipos de referências, tais como:

Tipo	Objeto
<code>article</code>	artigos em periódicos e revistas
<code>book</code>	livros publicados
<code>booklet</code>	livros sem editor ou patrocinador
<code>conference</code>	conferências e congressos
<code>inproceedings</code>	trabalhos publicados em conferências
<code>manual</code>	manuals técnicos
<code>masterthesis</code>	dissertações de mestrado
<code>pdhthesis</code>	teses de doutorado
<code>techreport</code>	relatórios técnicos
<code>misc</code>	outros tipos de publicação

- Depois do tipo, segue um grupo com os campos de informações sobre a obra

- Para cada tipo de obra, existem campos obrigatórios e campos opcionais. Por exemplo:

```
@article{tag ,  
  author   = "" ,  
  title    = "" ,  
  journal  = "" ,  
  %volume  = "" ,  
  %number  = "" ,  
  %pages   = "" ,  
  year     = "" ,  
  %month   = "" ,  
  %note    = "" ,  
}
```

@article

Campos obrigatórios:

author, title, journal, year.

Campos opcionais:

volume, number, pages,
month, note.

— **BibTeX** não permite comentários dentro das entradas, se precisar comente fora delas

- Existem ótimos aplicativos para auxiliar no preenchimento do arquivo **.bib**, inclusive com busca online de títulos e autores para preencher automaticamente os campos necessários. Teste algum⁷.

⁷Sugestão: <http://jabref.sourceforge.net/>

■ Como usar o BibTeX?

- (1) Criar o arquivo `documento.bib` e salvar na pasta do `documento.tex`
- (2) Incluir entradas para as obras consultadas em `documento.bib`
- (3) No final do `documento.tex`, antes do comando `\end{document}`, inserir os comandos:

```
\bibliographystyle{plain}  
\bibliography{documento}
```

- (4) Para fazer citação de uma obra, use o comando `\cite{tag-da-obra}`

■ Existem outros estilos além do `plain` para o BibTeX. Teste alguns deles ⁸

⁸<http://www.univie.ac.at/nuhag-php/bibtex/bibstyles.pdf>

Caixas

6 Caixas

- Idéia geral
- Minipáginas
- Caixas flutuantes
- Figuras
- Modificando o tamanho

- O comando `\mbox` permite ao autor criar caixas simples em torno de um texto. Isso proíbe o \LaTeX de separar seu conteúdo: caixas são unidades indivisíveis.

```
\mbox{Esta frase não será separada pelo \LaTeX.}
```

Atenção, se a caixa não couber numa linha o resultado será ruim

- O comando `\makebox` permite ao autor especificar a largura da caixa e a posição do texto dentro dela.

```
\makebox[8cm]{Este texto indivisível está centralizado.} \\
\makebox[8cm][c]{Este texto indivisível está centralizado.} \\
\makebox[8cm][s]{Este texto indivisível está justificado.}
```

A forma geral do comando é: `\makebox[largura][posição]{texto}`

O parâmetro posição é opcional e o padrão é centralizado. As opções são:

`[c]` = centralizado, `[l]` = à esquerda, `[r]` = à direita, `[s]` = justificado

- Dois outros comandos funcionam de modo análogo, mas acrescentando molduras nas caixas: `\fbox` e `\framebox`.

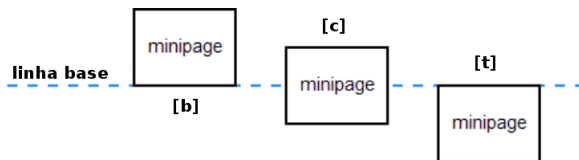
6.2. O ambiente minipage

(1/3)

- O ambiente `minipage` é muito útil e permite criar uma caixa que é uma minipágina com todas as características de uma página normal, isto é, pode ter cabeçalho, notas de rodapé etc.

```
\begin{minipage}[posição]{largura}  
    % conteúdo da minipágina  
\end{minipage}
```

O parâmetro `posição` é opcional e o padrão é centralizado. As opções são:
[c] = centralizado, **[b]** = linha do fundo, **[t]** = linha do topo



A posição especifica como fica o alinhamento da minipage na **linha base**

- O ambiente `minipage` pode ser utilizado para criar *layouts*

% Primeira minipágina

```
\begin{minipage}[t]{0.30\linewidth}
```

```
\color{red}
```

Esta minipágina contém um espaço equivalente à 30\% da largura da linha da página. Observe como o texto se ajusta ao espaço da minipágina.

```
\end{minipage}
```

```
\hspace{0.05\linewidth} % IMPORTANTE: sem linha em branco aqui
```

% Segunda minipágina

```
\begin{minipage}[t]{0.65\linewidth}
```

```
\color{blue}
```

Deste lado temos uma minipágina com espaço de apenas 65\% da largura da linha da página.

Dentro de uma minipágina podemos inserir o que quisermos, figuras, tabelas, etc. Isso permite, por exemplo, colocar textos e figuras lado a lado.

```
\end{minipage}
```

- Resultado da compilação do código do slide anterior:

Esta minipágina contém um espaço equivalente à 30% da largura da linha da página. Observe como o texto se ajusta ao espaço da minipágina.

Deste lado temos uma minipágina com espaço de apenas 65% da largura da linha da página. Dentro de uma minipágina podemos inserir o que quisermos, figuras, tabelas, etc. Isso permite, por exemplo, colocar textos e figuras lado a lado.

- O ambiente `minipage` na sua forma completa admite mais dois parâmetros:

```
\begin{minipage}[posição][A][P]{largura}
    % conteúdo da minipágina
\end{minipage}
```

Forma geral: `\begin{minipage}[posição][A][P]{largura}` onde:

A especifica a altura da minipage

P especifica o posicionamento vertical do texto dentro da minipage

- Uma **caixa flutuante** é uma unidade indivisível que contém textos, imagens, etc
 - ela não pode ser quebrada durante a mudança de página
 - ela é útil para tratar objetos que não se encaixam na página corrente
 - ela não faz parte do fluxo normal do texto e é posicionada diferentemente
- Flutuantes costumam incomodar muito os novatos com mentalidade **WYSIWYG**
- Flutuantes são adornados com legendas e recebem numeração automática, por isso compõem índices específicos disponibilizados automaticamente
- Exemplos típicos de flutuantes são **figuras** e **tabelas**. No decorrer do curso estudaremos esses flutuantes e veremos outros
- O autor também pode criar flutuantes (utilizando o pacote **float**) e alguns pacotes provêm outros além das **figuras** e **tabelas**

- As **caixas flutuantes** são especiais porque é o \LaTeX e não o autor que escolhe onde posicioná-las na página para alcançar o melhor efeito, o autor pode apenas sugerir as opções de posicionamento que mais lhe agradam.

```
\begin{tipoFlutuante}[posição]
    % conteúdo da caixa flutuante
\end{tipoFlutuante}
```

- As opções de posicionamento que o autor pode sugerir são:
 - [t]** no topo da página (atual ou seguinte)
 - [b]** no fundo da página (atual ou seguinte)
 - [p]** numa página especial para caixas flutuantes (mais à frente)
 - [h]** aqui (ou melhor, tão breve quanto possível a partir daqui)
 - [!]** força uma opção ignorando considerações do \LaTeX (cuidado!)
 - [H]** precisamente aqui, requer o pacote **float** (cuidado!)
- Como o autor não sabe ao certo onde será efetivamente posicionada uma caixa flutuante na página, é normal utilizar referências cruzadas para apontar para esses objetos ao longo do texto.

- O autor pode inserir figuras diretamente no texto, mas figuras são **caixas**

Figura `\includegraphics [scale=0.10]{knuth.png}` inserida no texto.



Figura inserida no texto.

- Inserir diretamente uma figura no texto pode não gerar bom resultado, então o \LaTeX oferece como alternativa o ambiente **figure** que cria uma caixa flutuante para posicionar figuras de forma adequada no documento

```
\begin{figure}[posição]  
... figura e descrição ...  
\end{figure}
```

- O ambiente **figure** é normalmente utilizado da forma abaixo

```
\begin{figure}[posição]  
  \includegraphics[opções]{arquivo}  
  \caption{legenda}  
  \label{tag}  
\end{figure}
```

- Opções de posicionamento **[posição]**: são as mesmas das caixas flutuantes
 - [t]** no topo da página (atual ou seguinte)
 - [b]** no fundo da página (atual ou seguinte)
 - [p]** numa página especial para caixas flutuantes (mais à frente)
 - [h]** aqui (ou melhor, tão breve quanto possível a partir daqui)
 - [!]** força uma opção ignorando considerações do LaTeX (cuidado!)
 - [H]** precisamente aqui, requer o pacote **float** (cuidado!)
- Marca para referência **{tag}**: criada pelo comando **\label** ⁹

⁹Ver seção 4. 4 sobre referências cruzadas.

- Apenas para comodidade, segue a forma do ambiente **figure** de novo:

```
\begin{figure}[posição]
  \includegraphics[opções]{arquivo}
  \caption{legenda}
  \label{tag}
\end{figure}
```

- Opções da figura **[opções]**: as mais comuns são
 - scale=xx** fator de escala (0.5 reduz o tamanho da figura pela metade)
 - height=xx** altura da imagem
 - width=xx** largura da imagem
 - angle=xx** ângulo de rotação (sentido anti-horário)
 - valign=x** alinhamento vertical (posição na linha base, opções: t, c, b) ¹⁰
- Legenda para a figura **{legenda}**: criada pelo comando **\caption** ¹¹
- Nome do arquivo **{arquivo}**: vários formatos são permitidos (png, jpg, etc)

¹⁰Requer **\usepackage[export]{adjustbox}** no preâmbulo

¹¹Se **\caption** vem antes de **\includegraphics**, a legenda vem antes da figura.

Exemplo:

```
\begin{itemize}
  \item A figura Fig~\ref{fig:knuth} apresenta uma foto
    antiga do criador do \TeX.
  \begin{figure}[H]
    \centering
    \includegraphics[scale=0.10]{knuth.png}
    \caption{\footnotesize Donald Knuth, Autor do TAOCP}
    \label{fig:knuth}
  \end{figure}

  \item A figura Fig~\ref{fig:knuth-rotacao} é semelhante à
    Fig~\ref{fig:knuth} porém com rotação de  $90^\circ$ , com 90\%
    do tamanho anterior.
  \begin{figure}[H]
    \centering
    \includegraphics[scale=0.09,angle=90]{knuth.png}
    \caption{Donald Knuth, Autor do TAOCP e \TeX}
    \label{fig:knuth-rotacao}
  \end{figure}
\end{itemize}
```

Resultado:

A figura Fig 1 apresenta uma foto antiga do criador do $\text{T}_{\text{E}}\text{X}$.



Figura 1: Donald Knuth, Autor do TAOCP

A figura Fig 2 é semelhante à Fig 1, porém com rotação de 90° , com 90% do tamanho anterior.



Figura 2: Donald Knuth, Autor do TAOCP e do $\text{T}_{\text{E}}\text{X}$

6.5. Modificando o tamanho da caixa

- O autor pode alterar o tamanho de uma caixa utilizando o comando `\resizebox{largura}{altura}{caixa}` ¹²
- Para modificar a largura da caixa, mantendo a proporção, use `\resizebox{largura}{!}{caixa}`
- Para modificar o tamanho da caixa por escala, mantendo a proporção, use `\scalebox{proporção}{caixa}` ¹²
- Exemplo:

```
\resizebox{!}{1.5cm}{$\bigstar$}
```



¹²Requer o pacote **graphicx** carregado.

Matemática

7 Matemática

- Idéia geral
- Símbolos e operadores
- Fórmulas
- Delimitadores
- Matrizes e arranjos
- Equações e teoremas

7.1. Notação matemática no \LaTeX

- Uma das maiores vantagens do \LaTeX consiste na possibilidade de criar fórmulas matemáticas com boa apresentação visual
- Fórmulas matemáticas são inseridas no \LaTeX por várias maneiras:
 - Ambiente **math** (ou $\$ \dots \$$), para expressões matemáticas *inline*
 - Ambiente **displaymath** (ou $\$\$ \dots \$\$$), para exibir equações
 - Ambiente **equation**, para equações numeradas
- Dentro destes ambientes é possível incluir facilmente:

■ operadores matemáticos	■ binômios
■ letras gregas	■ radiciação
■ notações em lógica	■ somatórios e produtórios
■ potenciação	■ integração e derivação
■ frações	■ matrizes e arranjos
■ funções	■ símbolos matemáticos

(1/2)

- Operadores matemáticos: + - = ! | () [] { }
- Alguns símbolos matemáticos dentre os muitos disponíveis:

Grupo	Sintaxe	Como é exibido
funções padrão	<code>\cos{x} + \ln{y} + \operatorname{sgn}{z}</code>	$\cos x + \ln y + \operatorname{sgn} z$
arit. modular	<code>s_k \equiv 0 \pmod{m}</code>	$s_k \equiv 0 \pmod{m}$
espaçamentos	<code>x \! x \, x \: x \; x \quad x \qquad x</code>	$x \! x \, x \: x \; x \quad x \qquad x$
derivação	<code>\nabla \quad \partial \quad \frac{\partial}{\partial x}</code>	$\nabla \quad \partial \quad \frac{\partial}{\partial x}$
conjuntos	<code>\exists x,y \quad \forall x \in \mathbb{N} \quad y \subseteq A \cup B \cap C</code>	$\exists x,y \quad \forall x \in \mathbb{N} \quad y \subseteq A \cup B \cap C$
lógica	<code>p \wedge \overline{q} \rightarrow p \vee \neg q</code>	$p \wedge \overline{q} \rightarrow p \vee \neg q$
raízes	<code>\sqrt{2} \approx 1{,}4 \quad \sqrt[n]{x}</code>	$\sqrt{2} \approx 1,4 \quad \sqrt[n]{x}$
op. relacionais	<code>\sim \simeq \cong \leq \geq \equiv \neq</code>	$\sim \simeq \cong \leq \geq \equiv \neq$
geometria	<code>\triangle \; \angle \; \perp \; 45^\circ</code>	$\triangle \; \angle \; \perp \; 45^\circ$
setas	<code>\leftarrow \; \rightarrow \; \leftrightarrows \; \longleftarrow \; \longrightarrow \; \nearrow \; \searrow \; \swarrow \; \nwarrow \; \uparrow \; \downarrow \; \updownarrow</code>	$\leftarrow \; \rightarrow \; \leftrightarrows \; \longleftarrow \; \longrightarrow \; \nearrow \; \searrow \; \swarrow \; \nwarrow \; \uparrow \; \downarrow \; \updownarrow$
especiais	<code>\infty \pm \mp \bullet \circ \oplus \otimes</code>	$\infty \pm \mp \bullet \circ \oplus \otimes$

7.2. Símbolos e operadores

(2/2)

■ Fontes úteis:

Sintaxe	Letra Grega
<code>\alpha</code>	α
<code>\beta</code>	β
<code>\gamma</code>	γ
<code>\delta</code>	δ
<code>\epsilon</code>	ϵ
<code>\zeta</code>	ζ
<code>\eta</code>	η
<code>\theta</code>	θ
<code>\kappa</code>	κ
<code>\lambda</code>	λ
<code>\mu</code>	μ
<code>\nu</code>	ν
<code>\xi</code>	ξ
<code>\pi</code>	π
<code>\rho</code>	ρ
<code>\sigma</code>	σ
<code>\tau</code>	τ
<code>\phi</code>	ϕ
<code>\chi</code>	χ
<code>\psi</code>	ψ
<code>\omega</code>	ω

Sintaxe	Letra Grega
<code>\Gamma</code>	Γ
<code>\Delta</code>	Δ
<code>\Theta</code>	Θ
<code>\Lambda</code>	Λ
<code>\Xi</code>	Ξ
<code>\Pi</code>	Π
<code>\Sigma</code>	Σ
<code>\Phi</code>	Φ
<code>\Psi</code>	Ψ
<code>\Omega</code>	Ω

Sintaxe	Letra ^a
<code>\mathbb{A}</code>	\mathbb{A}
<code>\mathbb{B}</code>	\mathbb{B}
<code>\mathbb{C}</code>	\mathbb{C}
<code>\vdots</code>	\vdots
<code>\mathbb{Z}</code>	\mathbb{Z}

^aRequer os pacotes `amsfonts` ou `amssymb`.

7.3. Fórmulas

(1/3)

■ Sobrescritos e subscritos

```
$f(x) = x^3 + 2x - 10$ \\
$a^{i+3}$ depois
$a_{i \times j}$ depois
$b_{2^3}$
```

$$f(x) = x^3 + 2x - 10$$

$$a^{i+3} \text{ depois } a_{i \times j} \text{ depois } b_2^3$$

■ Sobrelinhas e vetores

```
$_{\hat{a}} + \widehat{ghi} + \overline{b}$
depois
$_{\overrightarrow{ab}} + \overleftarrow{cd}$
```

$$\hat{a} + \widehat{ghi} + \overline{b} \text{ depois } \overrightarrow{ab} + \overleftarrow{cd}$$

■ Sobrechaves e subchaves

```
$_{\begin{matrix} 5050 \\ \overbrace{1+2+\cdots+100} \\ \end{matrix}}$
```

$$\overbrace{1+2+\cdots+100}^{5050}$$

```
$_{\begin{matrix} \\ \underbrace{a+b+\cdots+z} \end{matrix}}$
```

$$\underbrace{a+b+\cdots+z}$$

■ Somatórios

```


$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$


```

$$\sum_{k=1}^n k = \frac{n}{2}(n+1)$$

■ Produtórios

```


$$\prod_{i=1}^n x_i = n!$$


```

$$\prod_{i=1}^n x_i = n!$$

■ Limites

```


$$\lim_{n \rightarrow \infty} x_n$$


```

$$\lim_{n \rightarrow \infty} x_n$$

■ Integrais

```


$$\int_{-n}^n e^x \, dx \quad \oint x^3 \, dx$$


```

$$\int_{-n}^n e^x \, dx \quad \oint x^3 \, dx$$

■ Frações:

```
$$
\frac{-b \pm \sqrt{\Delta}}{2a}
$$
```

$$\frac{-b \pm \sqrt{\Delta}}{2a}$$

■ Coeficientes binomiais:

```
$$\{n \choose k\}
\quad
\quad
\binom{p}{x}$$
```

$$\binom{n}{k} \quad \binom{p}{x}$$

■ Funções por partes:

Seja $f(x)$ a função dada por $f(x) =$

```
\begin{cases}
0 & \& \text{se } x < 50 \\
2x^2 & \& \text{se } 50 \leq x < 300 \\
\sqrt{x + \cos\{x\}} & \& \text{se } x \geq 300.
\end{cases}
```

Seja $f(x)$ a função dada por $f(x) = \begin{cases} 0 & \text{se } x < 50 \\ 2x^2 & \text{se } 50 \leq x < 300 \end{cases}$

7.4. Delimitadores

- Chamamos **delimitadores** os sinais que envolvem objetos matemáticos
- São exemplos de delimitadores: parênteses, colchetes, chaves, barras verticais
- Para gerar delimitadores com tamanho ajustado na altura das caixas, o \LaTeX oferece os pares de comandos **left – right** :

Comando <i>left</i>	Comando <i>right</i>
<code>\left (</code>	<code>\right)</code>
<code>\left [</code>	<code>\right]</code>
<code>\left {</code>	<code>\right }</code>
<code>\left </code>	<code>\right </code>

Errado :

```
$$(\int_{0}^{\infty}\{e^{-st}\},dt)$$
```

$$(\int_0^\infty e^{-st} dt)$$

Certo :

```
$$\left(\int_{0}^{\infty}\{e^{-st}\},dt\right)$$
```

$$\left(\int_0^\infty e^{-st} dt\right)$$

- A base para construção das matrizes no \LaTeX é o ambiente `matrix`, onde as linhas são separadas por `\\` e as colunas são separadas por `&`.

```
$\begin{matrix}  
1 & 20 & 3 \\\br/>-2 & 9 & 14 \\\br/>15 & -23 & -32 \\\br/>\end{matrix}
```

1	20	3
-2	9	14
15	-23	-32

- Por padrão o ambiente `matrix` organiza por linhas com colunas centralizadas
- Para modificar o alinhamento por coluna, use uma opção de alinhamento¹³ ¹⁴

```
$\begin{matrix*}[r]  
1 & 20 & 3 \\\br/>-2 & 9 & 14 \\\br/>15 & -23 & -32 \\\br/>\end{matrix*}
```

1	20	3
-2	9	14
15	-23	-32

¹³As opções de alinhamento são as mesmas das tabelas: `[r]`, `[l]`, `[c]`. Ver seção 8

¹⁴Requer o pacote `mathtools`

- É simples cercar a matriz com algum par de delimitadores `left – right` ¹⁵
- Mas existem variantes do ambiente `matrix` que funcionam como atalhos:

```
$\begin{pmatrix}  
  a & b \\  
  c & d \\  
\end{pmatrix}
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

```
$\begin{vmatrix}  
  a & b \\  
  c & d \\  
\end{vmatrix}
```

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

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

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
$\begin{Vmatrix}  
  a & b \\  
  c & d \\  
\end{Vmatrix}
```

$$\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$$

```
$\begin{Bmatrix}  
  a & b \\  
  c & d \\  
\end{Bmatrix}
```

$$\begin{Bmatrix} a & b \\ c & d \end{Bmatrix}$$

- Quando tratando com matrizes de tamanho arbitrário, é comum usar elipses

```
$A_{m,n} =
\begin{pmatrix}
a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m,1} & a_{m,2} & \cdots & a_{m,n}
\end{pmatrix}
```

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

- Quando tratando com frações, pode ocorrer de faltar espaço na linha

Resultado
ruim

```
\begin{bmatrix}
\frac{5}{6} & 0 \\
\frac{5}{6} & 0
\end{bmatrix}
```

$$\begin{bmatrix} \frac{5}{6} & 0 \\ \frac{5}{6} & 0 \end{bmatrix}$$

Problema
corrigido

```
\begin{bmatrix}
\frac{5}{6} & 0 \\
\frac{5}{6} & 0
\end{bmatrix} \ll[2mm]
```

$$\begin{bmatrix} \frac{5}{6} & 0 \\ \frac{5}{6} & 0 \end{bmatrix} \ll[2mm]$$

- Às vezes é necessário ter um controle fino sobre as linhas e colunas da matriz. Isso pode ser obtido com o uso do ambiente `array` que é essencialmente uma versão modo-matemático para o ambiente `tabular`¹⁶

Exemplo:

```
$\begin{array}{c|c|c|c}
1 & 2 & 3 & \\
\hline
4 & 5 & 6 & \\
\hline
7 & 8 & 9 & \\
\end{array}$
```

1	2	3
4	5	6
7	8	9

- Para mais detalhes, estude o funcionamento do ambiente `tabular`

¹⁶Ver seção 8

- O ambiente `equation` numera automaticamente a equação:

```
A parábola é um tipo de polinômio.  
\begin{equation}  
  \label{eq:parabola}  
   $f(x) = ax^2 + bx + c$   
\end{equation}  
A Equação~\eqref{eq:parabola} é um  
exemplo.
```

A parábola é um tipo de polinômio.

$$f(x) = ax^2 + bx + c \quad (1)$$

A Equação (1) é um exemplo.

- Com os comandos `\ref{...}` ou `\eqref{...}` é possível referenciar a equação no texto. A diferença entre esses comandos é que o primeiro produz “1” e o segundo “(1)”
- Para usar o comando `\eqref` é preciso incluir o pacote `amsmath`

- Para numerar equações subordinadas, use o ambiente **subequations** :

```
\begin{subequations}
Equações de Maxwell:
\begin{align}
B' &= -\nabla \times E \\
E' &= \nabla \times B - 4\pi j
\end{align}
\end{subequations}
```

Equações de Maxwell:

$$B' = -\nabla \times E \quad (2a)$$

$$E' = \nabla \times B - 4\pi j \quad (2b)$$

- Um `\label{...}` pode ser incluído no final de cada linha, antes do `\\`
- Existem vários outros comandos para formatar equações, como:

Comandos	Descrição
<code>\overset</code> e <code>\underset</code>	Posiciona símbolos acima e abaixo na equação
<code>\xLeftarrow[under]{over}</code> ¹⁷	Posiciona símbolos acima e abaixo de setas

¹⁷ Este comando e seus análogos: `\xrightarrow` etc

- Equações numeradas e alinhadas em múltiplas linhas:

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

$$(a+b)^2 = (a+b)(a+b) \tag{3}$$

$$= a(a+b) + b(a+b)$$

$$= a^2 + 2ab + b^2 \tag{4}$$

- O comando `\nonumber` cancela uma numeração

- Equações não numeradas e alinhadas em múltiplas linhas:

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

$$\begin{aligned}(a+b)^2 &= (a+b)(a+b) \\ &= a(a+b) + b(a+b) \\ &= a^2 + 2ab + b^2\end{aligned}$$

- O comando `\newtheorem` permite ao autor criar novos ambientes para numerar definições, teoremas, provas etc
- O comando para criar o novo ambiente é posicionado no preâmbulo

```
\newtheorem{teorema}{Poderoso Teorema}
```

- Depois de criado, o novo ambiente pode ser usado no corpo do documento:

```
\begin{teorema}  
  Seja  $f(x)$  uma função cuja derivada existe em todo ponto, então  $f(x)$   
  é uma função contínua.  
\end{teorema}
```

Poderoso Teorema 1 *Seja $f(x)$ uma função cuja derivada existe em todo ponto, então $f(x)$ é uma função contínua.*

Tabelas

8 Tabelas

- Idéia geral
- Colunas formatadas
- Tabelas coloridas
- Família de pacotes

- O autor pode inserir tabelas diretamente no texto, mas tabelas são caixas

```
Tabela \begin{tabular}{|c|c|}
\hline bit & bool \\
\hline 0 & false \\
\hline 1 & true \\
\hline
\end{tabular} inserida no texto.
```

Tabela

bit	bool
0	false
1	true

inserida no texto.

- Inserir diretamente uma tabela no texto pode não gerar bom resultado, então o \LaTeX oferece como alternativa o ambiente `table` que cria uma caixa flutuante para posicionar tabelas de forma adequada no documento

```
\begin{table}[posição]
... tabela e descrição ...
\end{table}
```

- O ambiente `tabular` cria a tabela
- O ambiente `table` cria um flutuante para encapsular a tabela
- A complexidade do ambiente `tabular` exige uma boa indentação

- O ambiente `table` é normalmente utilizado da forma abaixo

```
\begin{table}[posição]
... ambiente tabular aqui ...
\caption{legenda}
\label{tag}
\end{table}
```

- Opções de posicionamento **[posição]**: são as mesmas das caixas flutuantes

- [t]** no topo da página (atual ou seguinte)
- [b]** no fundo da página (atual ou seguinte)
- [p]** numa página especial para caixas flutuantes (mais à frente)
- [h]** aqui (ou melhor, tão breve quanto possível a partir daqui)
- [!]** força uma opção ignorando considerações do LaTeX (cuidado!)
- [H]** precisamente aqui, requer o pacote `float` (cuidado!)

- Legenda para a tabela **{legenda}**: criada pelo comando `\caption` ¹⁸

- Marca para referência **{tag}**: criada pelo comando `\label` ¹⁹

¹⁸Se `\caption` vem antes do ambiente `tabular`, a legenda vem antes da tabela.

¹⁹Ver seção 4. 4 sobre referências cruzadas.

- O ambiente `tabular` tem a forma geral:

```
\begin{tabular}[posição]{  
especificação  
... conteúdo da tabela ...  
\end{tabular}
```

- Alinhamento vertical na linha base **[posição]**: parâmetro opcional
 - opções: **[t]**, **[c]**, **[b]**
 - normalmente você **não** precisa desse parâmetro
- Especificação das colunas e linhas verticais **{especificação}**:
 - informa o número de colunas na tabela e a formatação para cada coluna
 - o número de colunas é inferido automaticamente dos argumentos
 - a largura de cada coluna é determinada automaticamente
 - a existência de uma linha vertical é indicada por `|`

- Os símbolos que descrevem o formato de uma coluna na **{especificação}** são:

Coluna	Descrição
l	coluna com conteúdo justificado à esquerda
c	coluna com conteúdo centralizado
r	coluna com conteúdo justificado à direita
p{largura}	coluna com parágrafo alinhado verticalmente no topo
m{largura}	coluna com parágrafo alinhado verticalmente no centro ²⁰
b{largura}	coluna com parágrafo alinhado verticalmente no fundo ²⁰
	linha vertical simples
	linha vertical dupla

- O \LaTeX por padrão não controla quebra de linha na tabela, por isso a largura das colunas pode estourar o tamanho da linha. Para contornar esse problema, são ofertadas as opções **p**, **m**, **b** que permitem especificar a largura da coluna
- A largura da coluna pode ser indicada com uma unidade válida ²¹ ou como proporção a `\textwidth`.

²⁰Requer o pacote `array` carregado.

²¹Por exemplo, **cm** ou **pt**.

A Tabela~\ref{tab:tabuada-and} mostra a tabuada para o operador lógico \wedge :

```
\begin{table}[h]
\centering
\begin{tabular}{|cc|c|}
\hline
 $p$  &  $q$  &  $p \wedge q$  \\
\hline
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\hline
\end{tabular}
\caption{Tabela verdade para  $(p \wedge q)$ }
\label{tab:tabuada-and}
\end{table}
```

A Tabela 1 mostra a tabuada para o operador lógico \wedge :

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 1: Tabela verdade para $(p \wedge q)$

O ambiente **tabular** contém as linhas da tabela:

- As colunas são separadas por **&**
- As linhas são separadas por ****
- **\newline** começa nova linha em parágrafo coluna
- **\hline** linha horizontal
- **\cline{i-j}** linha horizontal parcial da coluna i até j

8.2. Personalizando colunas

- O conteúdo de uma coluna pode ser formatado com as macros $\gt\{...\}$ e $\lt\{...\}$
- Na **{especificação}** da tabela, as colunas personalizadas tem a forma:

$\gt\{$ comandos-antes $\}$ formato $\lt\{$ comandos-depois $\}$

- A macro $\gt\{...\}$ indica comandos para executar **no início** de cada célula da coluna
- A macro $\lt\{...\}$ indica comandos para executar **no final** de cada célula da coluna

Exemplos

$\gt\{\$ \} p\{4cm\} \lt\{\$ \}$

Coluna de 4cm em modo matemático

$\gt\{\backslash bgroup\backslash tiny\backslash textcolor\{red\}\} r \lt\{\backslash egroup\}$

Coluna com texto pequeno em vermelho à direita

8.3. Tabelas coloridas

- Para colorir tabelas, inclua no preâmbulo: `\usepackage[table]{xcolor}`
- Para colorir uma célula da tabela use, no início da célula:

```
\cellcolor{cor}
```

- Para colorir uma linha da tabela use, no início da linha:

```
\rowcolor{cor}
```

- Para colorir as linhas alternando cores use, antes do ambiente **tabular** :

```
\rowcolors{linha-inicial}{cor-linhas-ímpares}{cor-linhas-pares}
```

Exemplos

```
\rowcolors{2}{green}{pink}  
\begin{tabular}{|ccc|}  
 \hline \rowcolor{blue!30}  
 impar & a & b \\  
 \hline  
 par & c & d \\  
 impar & e & f \\  
 par & g & h \\  
 \hline
```

impar	a	b
par	c	d
impar	e	f
par	g	h

8.4. Alguns pacotes relacionados com tabelas (1/4)

Pacote	Descrição breve
tabularx	cria tabela com largura definida, mas mantendo a definição automática da largura das colunas
booktab	cria tabela elegante com estilo semelhante ao usado em muitos livros
longtable	cria tabela que pode se alongar e continuar além de uma página

- Esses pacotes definem variantes para o ambiente `tabular`
- Geralmente usam a forma geral do ambiente `tabular` e acrescentam alguma funcionalidade específica
- Vamos comentar a idéia geral, para mais detalhes ver documentação do pacote

- Inclua no preâmbulo: `\usepackage{tabularx}`
- A forma geral exige o parâmetro adicional **{largura}**:

```
\begin{tabularx}[posição]{largura}{especificação}  
... conteúdo da tabela ...  
\end{tabularx}
```

- A **{largura}** define a largura total da tabela
- A **{especificação}** aceita as opções de **tabular**
- A **{especificação}** aceita a opção adicional **X**
- A opção **X** faz o compilador calcular automaticamente a largura da coluna sem ultrapassar o largura definida para a tabela

- Inclua no preâmbulo: `\usepackage{booktab}`
- O pacote **booktab** não muda o ambiente **tabular**, a forma geral do ambiente permanece a mesma:

```
\begin{tabular}[posição]{especificação}  
... conteúdo da tabela ...  
\end{tabular}
```

- Na descrição do conteúdo da tabela, o comando `\hline` é substituído por
 - `\toprule` para a primeira linha da tabela
 - `\bottomrule` para a última linha da tabela
 - `\midrule` para as linhas entre a primeira e a última
- Quando usar as linhas definidas em **booktab**, não use linhas verticais

- Inclua no preâmbulo: `\usepackage{longtable}`
- Use o ambiente `longtable` cuja forma geral é semelhante a `tabular` :

```
\begin{longtable}{especificação}  
... conteúdo da tabela ...  
\end{longtable}
```

Algoritmos

9 Algoritmos

- Idéia geral
- Pacote listings
- Pacote algorithmic
- Pacote algorithmicx
- Pacote algorithm2e

9.1. Algoritmos no \LaTeX

- Existem vários pacotes para imprimir algoritmos de forma elegante no \LaTeX . Esses pacotes oferecem comandos que formatam as principais construções dos algoritmos (laços, condicionais, etc)
- É necessário escolher qual pacote utilizar, porque existem diferenças e alguns são incompatíveis entre si
- Vamos apresentar a idéia geral e mostrar exemplos de uso dos principais, para maiores detalhes consulte a documentação do pacote

Pacote	Descrição breve
listings	Este pacote difere dos outros por não oferecer comandos de formatação, mas é muito útil provendo um ambiente adequado para apresentação de códigos fontes
algorithm	Define o ambiente <code>algorithm</code> , uma caixa flutuante para apresentação de algoritmos
algorithmic ¹	Define o ambiente <code>algorithmic</code> e comandos para apresentação de pseudo-códigos ²
algorithmicx	Este pacote não define comandos para apresentação, mas facilita a criação deles ² . É usado junto com algum <i>layout</i> de comandos, use o pacote <code>algpseudocode</code>
algorithm2e ¹	Pacote sofisticado com customizações para apresentação de pseudo-códigos ²

¹ Pacote não compatível com o muito usado pacote `revtex4-1`

² Os pacotes `algorithmic`, `algorithmicx`, `algorithm2e` são incompatíveis entre si

- Oferece um ambiente para apresentação elegante de códigos fontes, facilitando sua leitura ao destacar os comandos da linguagem escolhida
- Para utilizar, inclua no preâmbulo: `\usepackage{listings}`
- Para inserir código fonte no próprio documento:

```
\begin{lstlisting}  
... escreva o código fonte aqui ...  
\end{lstlisting}
```

- Para inserir código armazenado em arquivo fonte de uma dada linguagem:

```
\lstinputlisting[language=C]{arquivo.c}
```

- Para inserir código fonte em linha no próprio documento:

```
\lstinline !código  
fonte !
```

- ▶ é possível usar a alternativa `\lstinline{código fonte}`
- ▶ mas código fonte costuma conter `}`
- ▶ a solução é uma macro TeX onde `!` é um delimitador
- ▶ é possível escolher outro delimitador, por exemplo, `$`

- É possível personalizar a apresentação dos códigos fontes alterando a configuração padrão oferecida pelo pacote
- O exemplo abaixo mostra como alterar os parâmetros do pacote no preâmbulo:

```
\lstset{
  language=Java,                % linguagem do código
  otherkeywords={...},          % se quiser adicionar mais palavras-chave além do
    padrão
  backgroundcolor=\color{white}, % cor de fundo; requer \usepackage{xcolor}
  basicstyle=\footnotesize,     % tamanho da fonte
  frame=single,                 % usa moldura simples em volta do código
  rulecolor=\color{black},      % cor da moldura
  keepspaces=true,              % conserva espaços no texto, útil para indentação
  keywordstyle=\color{blue},    % estilo para as palavras-chave
  commentstyle=\color{green},   % estilo para os comentários
  stringstyle=\color{purple},   % estilo para strings literais
  numbers=left,                 % posição da numeração das linhas; opções: none,
    right
  numbersep=5pt,                % espaço entre a numeração das linhas e o código
  numberstyle=\tiny\color{gray}, % estilo para a numeração das linhas
  showspaces=false,             % usa marca especial para mostrar espaços ?
  showstringspaces=false,       % usa marca especial para mostrar espaços em strings
    ?
  showtabs=false,               % usa marca especial para mostrar tabs em strings ?
  stepnumber=2,                 % passos entre linhas numeradas; 1 numera todas
    linhas
  tabsize=2,                    % tamanho do tab em espaços
}
```

- O pacote também permite definir estilos para uso posterior. Os exemplos abaixo ilustram isso:

```
\lstdefinestyle{customc}{  
  language=C, frame=L, xleftmargin=\parindent, breaklines=true, showstringspaces=  
  false,  
  basicstyle=\footnotesize\ttfamily, keywordstyle=\bfseries\color{green!40!black},  
  commentstyle=\itshape\color{purple!40!black}, identifierstyle=\color{blue},  
  stringstyle=\color{orange},  
}  
  
\lstdefinestyle{customasm}{  
  language=[x86masm]Assembler, frame=L, xleftmargin=\parindent,  
  basicstyle=\footnotesize\ttfamily, commentstyle=\itshape\color{purple!40!black},  
}  
  
\lstset{style=customc} % define o estilo padrão
```

- Uma vez definidos os estilos, será simples invocá-los nos comandos:

```
\lstinputlisting[style=customasm]{arquivo.asm}
```


- Oferece um ambiente para apresentação elegante de pseudo-códigos
- Oferece os seguintes comandos de formatação:

```
\STATE <text>
\IF{<condition>} <text> \ENDIF
\IF{<condition>} <text> \ELSE <text> \ENDIF
\IF{<condition>} <text> \ELSIF{<condition>} <text> \ELSE <text> \ENDIF
\FOR{<condition>} <text> \ENDFOR
\FORALL{<condition>} <text> \ENDFOR
\WHILE{<condition>} <text> \ENDWHILE
\REPEAT <text> \UNTIL{<condition>}
\LOOP <text> \ENDLOOP
\REQUIRE <text>
\ENSURE <text>
\RETURN <text>
\PRINT <text>
\COMMENT{<text>}
\AND, \OR, \XOR, \NOT, \TO, \TRUE, \FALSE
```

- Para usar, inclua no preâmbulo:

```
\usepackage{algorithm}
\usepackage{algorithmic}
```

- ```

\begin{algorithm}[H]
 \alsetup{linenosize=\tiny}
 \caption{Procedimiento de
Euclides}
 \label{alg:euclides}
 \begin{algorithmic}[1]
 \REQUIRE a, b
 \ENSURE $\gcd(a,b)$
 \STATE $r \gets a \bmod b$
 \WHILE{$r \neq 0$}
 \STATE $a \gets b$
 \STATE $b \gets r$
 \STATE $r \gets a \bmod b$
 \ENDWHILE
 \RETURN b
 \end{algorithmic}
\end{algorithm}

```

```

1: $r \leftarrow a \bmod b$
2: while $r \neq 0$ do
3: $a \leftarrow b$
4: $b \leftarrow r$
5: $r \leftarrow a \bmod b$
6: end while
7: return b

```

- ```
\floatname{algorithm}{Procedure}
\newcommand{\algorithmicrequire}{\textbf{Input:}} % require
\newcommand{\algorithmicensure}{\textbf{Output:}} % ensure
```

- Este pacote facilita a criação de comandos de formatação e normalmente é usado em conjunto com algum *layout*²² já codificado.
- O **algpseudocode** oferece os seguintes comandos de formatação:

```
\State <text>
\If{<condition>} <text> \EndIf
\If{<condition>} <text> \Else <text> \EndIf
\If{<condition>} <text> \ElseIf{<condition>} <text> \Else <text> \EndIf
\For{<condition>} <text> \EndFor
\ForAll{<condition>} <text> \EndFor
\While{<condition>} <text> \EndWhile
\Repeat <text> \Until{<condition>}
\Loop <text> \EndLoop
\Require <text>
\Ensure <text>
\Return <text>
\Comment{<text>}
\Statex % comando vazio, para indentação, gera linha em branco
\Procedure<name>{<params>} <text> \EndProcedure
\Function{<name>}{<params>} <text> \EndFunction
```

- Para usar, inclua no preâmbulo:

```
\usepackage{algorithm}
\usepackage{algpseudocode}
```

- Este pacote permite redefinir os comandos, mais detalhes na documentação

```

\begin{algorithm}[H]
  \caption{Procedimento de
  Euclides}
  \label{alg:euclides}
  \begin{algorithmic}[1]
    \Procedure{Euclides}{$a,b$}
      \Comment{Calcula o mdc(a,b)}
    \State $\textcolor{blue}{r} \textcolor{blue}{\gets} a \textcolor{blue}{\bmod} b$
    \While{$\textcolor{blue}{r} \textcolor{blue}{\neq} 0$}
      \State $\textcolor{blue}{a} \textcolor{blue}{\gets} b$
      \State $\textcolor{blue}{b} \textcolor{blue}{\gets} r$
      \State $\textcolor{blue}{r} \textcolor{blue}{\gets} a \textcolor{blue}{\bmod} b$
    \EndWhile
    \State \Return $b$
    \Comment{O mdc(a,b) está em
    $b$}
  \EndProcedure
\end{algorithmic}
\end{algorithm}

```

Algoritmo 1 Procedimento de Euclides

```

1: procedure EUCLIDES( $a, b$ ) ▷ Calcula o mdc(a,b)
2:    $r \leftarrow a \bmod b$ 
3:   while  $r \neq 0$  do
4:      $a \leftarrow b$ 
5:      $b \leftarrow r$ 
6:      $r \leftarrow a \bmod b$ 
7:   end while
8:   return  $b$  ▷ O mdc(a,b) está em  $b$ 
9: end procedure

```

- Para renomear as apresentações criadas pelos comandos:

```

\floatname{algorithm}{Algoritmo}
\renewcommand{\algorithmicrequire}{\textbf{Input:}} % require
\renewcommand{\algorithmicensure}{\textbf{Output:}} % ensure

```

■ Este pacote é mais sofisticado e exige mais esforço para usar bem:

- todo texto é tratado com um potencial comando alto-nível
- todo comando alto-nível termina com o fim-de-comando `\;`
- existem comandos de formatação que são predefinidos
- existem macros para o autor criar novos comandos de formatação
- existem comandos de configuração para o pacote

■ Existem muitos comandos de formatação predefinidos, seguem alguns:

- `\KwData{ input }, \KwResult{ output }, \KwIn{ input }, \KwOut{ output }`
- `\KwTo`
- `\KwReturn{ value }`
- `\Begin{ block inside }`
- `\If{ condition }{ then block }, \elif{ condition }{ then block } else block }`
- `\ul{ condition }{ then block }, \uElse{ condition }{ more one block } \Else{ last block }`
- `\While{ condition }{ text loop }, \Repeat{ end condition }{ text loop }`
- `\For{ condition }{ text loop }, \ForEach{ condition }{ text loop }, \ForAll{ condition }{ text loop },`

■ Existem muitos comandos para comentários, mas esse costuma atender bem:

- `\tcp*[pos]{ comment }` gera um comentário no estilo C++
- o parâmetro `[pos]` indica o alinhamento na linha, opções = `(r|l)`
- dica: se usar comentário na linha, não use o fim-de-comando `\;`

■ Existem muitas macros para criação de comandos, alguns exemplos:

- `\SetKw{KwAnd}{and}` define o comando `\KwAnd` que produz **and**
- `\SetKwFunction{DoThat}{Do}` define o comando `\DoThat{arg}` que produz **Do(arg)**
- `\SetKwInParam{Func}{(){}}` define a macro `\Func{ name }{ arg }` cuja `\Func{ function }{ arg1, arg2 }` produz **function(arg1, arg2)**

■ Existe uma enorme variedade de comandos e opções para customização

■ O apresentado aqui é suficiente para usar os recursos básicos do pacote

■ Para maiores detalhes consulte a documentação do pacote

- Difere dos pacotes anteriores pela enorme variedade de customizações

```
\begin{algorithm}[H]
\KwData{aula e slides}
\KwResult{aprendizado}
comece a estudar e praticar\;
\While{não terminou o curso}{
  estude um tópico\;
  \elf{entendeu o tópico}{
    \elf{era o último tópico da seção}{
      vá para a próxima seção\;
    }{ vá para o próximo tópico\;
    }
  }{ procure ajuda para eliminar a
  dúvida\;
  }
} % while
\caption{Como aprender LaTeX no curso}
\end{algorithm}
```

Algorithm 1: Como aprender LaTeX no curso

Data: aula e slides
Result: aprendizado

- 1 comece a estudar e praticar;
- 2 **while** não terminou o curso **do**
- 3 estude um tópico;
- 4 **if** entendeu o tópico **then**
- 5 **if** era o último tópico da seção **then**
- 6 vá para a próxima seção;
- 7 **else**
- 8 vá para o próximo tópico;
- 9 **end**
- 10 **else**
- 11 procure ajuda para eliminar a dúvida;
- 12 **end**
- 13 **end**

- Este código foi compilado com as opções de pacote no preâmbulo:

```
\usepackage{algorithm}
\usepackage[ruled, % cabeçalho em cima, entre duas linhas
linesnumbered, % linhas de código numeradas
lined          % linhas verticais de indentação
]{algorithm2e}
```

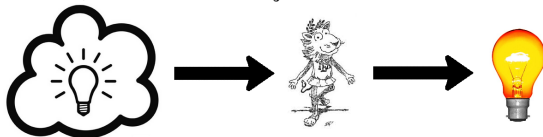
Gráficos

10 Gráficos

- Idéia geral
- Pacote tikz

10.1. Gráficos no \LaTeX

- Criar gráficos no \LaTeX pode consumir tempo, mas o resultado é ótimo
- Uma vez dominada a técnica, o trabalho de tradução idéia \rightarrow documento é otimizado



- Existem vários pacotes que suportam a criação de gráficos dentro do \LaTeX . Alguns deles:

Pacote	Descrição breve
<code>picture</code>	Pacote minimalista, oferece poucos recursos, mas produz figuras de poucos bytes
<code>pstricks</code>	Extensão poderosa do <code>picture</code> , é voltado para PostScript e não trabalha com pdflatex
<code>metapost</code>	Talvez o mais poderoso pacote, inspirado na Metafont , outra linguagem de descrição idealizada pelo Knuth, mas também é voltado para o PostScript
<code>tikz</code>	Pacote escolhido para ser tratado neste curso, será assunto da próxima seção

- O **PGF/TikZ**, doravante apenas **tikz**, é um sistema de duas camadas:
 - PGF** (*portable graphics format*) é a camada *back end* que oferece os comandos básicos para a construção dos gráficos
 - TikZ** é a camada *front end* que torna fácil o uso do **PGF**, é resultado de um casamento feliz de comandos inspirados na **Metafont** com um mecanismo de sintaxe inspirado no **pstricks**
- O **tikz** produz gráficos que são portáteis para Pdf e PostScript
- Outros pacotes focam no “como” desenhar, a filosofia do **tikz** permite isso, mas também permite trabalhar em nível mais alto, focando no “que” desenhar. Existem muitos subpacotes especializados para: autômatos, circuitos elétricos, mapas mentais, figuras geométricas etc
- Visite a página <http://www.texample.net/tikz/examples/>

- Para usar o `tikz`, inclua no preâmbulo:

```
\usepackage{tikz} % o pacote pgf será incluído automaticamente
\usetikzlibrary{ <lista de subpacotes separados por vírgulas> }
```

- Existem vários subpacotes (*libraries*) para o `tikz`, alguns exemplos:

“arrows”, “automata”, “backgrounds”, “calendar”, “chains”,
“matrix”, “mindmap”, “patterns”, “petri”, “shadows”,
“spy”, “trees”, “shapes.geometric”, “shapes.misc”

- Para desenhar, coloque os comandos no ambiente `tikzpicture`:

```
\begin{tikzpicture}[opções]
... comandos tikz ...
\end{tikzpicture}
```

ou, alternativamente, use o comando em linha:

```
\tikz[opções]{ ... comandos tikz ... }
```

- Muitas vezes o conceito de **grafo** facilita o entendimento no **tikz** :
 - o desenho é formado de vértices (**nodes**) conectado por arestas (**edges**)
 - uma sequência de vértices define um caminho (**path**)
 - formas prontas podem ser colocadas sobre os vértices
 - vértices e arestas podem ser decorados e/ou estilizados
- Os subpacotes importados fornecem diversos tipos de formas prontas
- Há dois modos básicos para posicionamento no desenho:
 - Modo absoluto: são fornecidas coordenadas

```
(1cm,2pt) % coord. cartesianas: 2cm no eixo x, 2pt no eixo y
(30:1cm) % coord. polar: 30 graus, 1cm de distancia do centro
(1,3)    % sem unidade, o padrao é cm
```

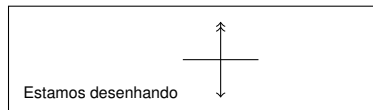
- Modo relativo: são fornecidas orientação relativas a elementos já posicionados

```
(1,0)      % ponto referencial
+(2,3)     % coord (2+1,3+0) e o ponto (1,0) continua sendo referencial
++(4,5)    % coord (4+1,5+0) e o ponto (5,5) torna-se referencial
```

Existem outros tipos de orientação relativa, por exemplo: *above*, *below*, *left*, *right*. No modo relativo frequentemente é estabelecida uma grade e as orientações então incluem uma distância. Por exemplo: *2 above* (duas posições acima)

- O desenho gerado pelo `tikz` é uma **caixa** ²³
- Exemplo:

```
Estamos desenhando
\begin{tikzpicture}
  \draw      (-0.25,0) --
(0.25,0);
  \draw[<->] (0,-0.25) --
(0,0.25);
\end{tikzpicture}
```



Observações:

- o comando `\draw`, como vários outros no `tikz`, tem a forma

```
\comando[opcoes] <ponto> <elemento>
                  <ponto> <elemento>
                  :
                  <ponto> <elemento>;
```

- o comando `\draw` mandou desenhar
- o elemento foi identificado, pelo `--`, como um caminho
- e o `tikz` desenhcou uma aresta até o outro ponto

²³Ver Seção 6

- O `tikz` oferece muitos recursos e dificilmente alguém aprenderá tudo dele
- Um bom caminho para iniciar o aprendizado é estudar os tutoriais do manual [2]
`www.texample.net/media/pgf/builds/pgfmanualCVS2012-11-04.pdf`
- Outra opção para iniciar, consultar o wikibooks do \LaTeX [5]
`https://en.wikibooks.org/wiki/LaTeX/PGF/TikZ`
- Uma boa prática, quando for desenhar no \LaTeX , é examinar um exemplo pronto daquilo que você precisa. Existem bons exemplos de uso do `tikz` no site
`http://www.texample.net/tikz/examples/`
- Felizmente o uso do `tikz` costuma ser intuitivo e com alguma prática ele será dominado naquilo que você precisa
- Não tente aprender tudo de uma vez, pegue um exemplo pronto e adapte para sua necessidade

Programação

11 Programação

- Idéia geral
- Pacotes de apoio
- Pacote xargs
- Pacote xparse
- Pacote environ
- Pacote xkeyval
- Pacote etoolbox

- Algumas vezes você precisa de um comando ou ambiente que não existe, nesse momento o \LaTeX fica ainda mais interessante:
 - ele permite você criar novos comandos e novos ambientes de acordo com sua necessidade
 - ele permite você utilizar comandos básicos de programação (condicionais e repetições)
 - se ainda não bastar, você pode apelar para o \TeX (não abordamos o \TeX neste curso)
- Vários pacotes que dão suporte para a programação dentro do \LaTeX . Alguns deles:

Pacote	Descrição breve
calc	Permite executar cálculos básicos com contadores e tamanhos
calculator	Permite usar uma calculadora dentro do \LaTeX
ifthen	Implementa o comando condicional
multido , pgffor	Pacotes concorrentes que implementam o comando de repetição O multido foi criado para o pstricks , mas pode ser usado sozinho O pgffor foi criado para o tikz , mas pode ser usado sozinho
etoolbox	Pacote de ferramentas completo, substitui muitos pacotes
environ	Implementa macros poderosas para criação de novos ambientes
xargs , xparse	Pacotes concorrentes, implementam macros para tratamento de parâmetros
xkeyval	Permite criar opções nomeadas (do tipo [key₁=val₁, key₂=val₂]) para comandos
xstring	Permite executar operações com strings

- Para criar um novo comando, use:

```
\newcommand{\nome}[num]{definição}
```

Cria o comando **\nome** com **num** parâmetros, acessados na **definição** assim:
#1 para o primeiro, **#2** para o segundo, etc

```
\newcommand{\aspas}[1]{'#1'} %... para criar
\aspas{teste} %..... para usar
```

- Para redefinir um comando existente, use:

```
\renewcommand{\comando}[num]{definição}
```

- Para criar um comando com um parâmetro opcional, use:

```
\newcommand{\nome}[num][padrão]{definição}
```

Cria o comando **\nome** com **num** parâmetros, o primeiro opcional, os outros não
 Se o parâmetro opcional não aparecer, ele assume o valor **padrão**

```
\newcommand{\moeda}[2][R]{#1\$, #2.00} %... para criar

\moeda{3} %..... para usar ..... vai gerar R$ 3.00
\moeda[US]{4} %..... para usar ..... vai gerar US$ 4.00
```

- Para criar um novo ambiente, use:

```
\newenvironment{nome}[num]{ antes }{ depois }
```

Cria o ambiente **nome** com **num** parâmetros, de modo análogo ao que ocorre na criação de comandos
Como se trata de um ambiente:

- 1► quando o comando **\begin{ nome }** é encontrado, o parâmetro **antes** é processado
- 2► o corpo do ambiente é processado
- 3► quando o comando **\end{ nome }** é encontrado, o parâmetro **depois** é processado

```
%--- para criar
\newenvironment{atencao}
{ \noindent \mbbox\bgroup \color{red} $\blacktrianglerightright$ }
{ $\blacktriangleleft$ \egroup }

%--- para usar      (digite o exemplo e compile para ver o resultado)
\begin{atencao}
  Exemplo de Uso
\end{atencao}
```

- Para executar o exemplo, inclua o pacote **amssymb**
- Para redefinir um ambiente existente, use **\renewenvironment** e proceda como no caso dos comandos
- Para criar um ambiente com um parâmetro opcional, proceda como no caso dos comandos

- Agora um exemplo mais complexo:

```
% Criando o novo ambiente
\newenvironment{moldura}[1] % moldura com titulo
{ %----- antes
\newcommand{\titmold}{#1}
\begin{tikzpicture}[scale=1]
\tikzstyle{titulo}=[anchor=north west,
text width={3cm}, inner sep={2mm},
fill={green!30}, draw={black}, text centered ]
\tikzstyle{texto}=[anchor=north west,
text width={0.9\textwidth}, inner sep={2mm},
fill={green!10}, draw={black}, text justified ]
\draw (0,0) node [style=texto]
\bgroup }
{ %----- depois
\egroup ;
\draw (1, 18pt) node [style=titulo] {\titmold} ;
\end{tikzpicture} }
```

Digite o exemplo e compile para ver o resultado

```
% Usando o novo ambiente
\begin{moldura}{Dica}
Pratique o LaTeX e em pouco tempo vai se acostumar com ele.
\end{moldura}
```

- Os **comandos frágeis** não funcionam sempre como você queria que fosse
 - ▶ Tecnicamente, quando o \TeX recebe um token, ele o expande e o executa (se possível)
 - A fase de expansão, dependendo do caso, poderia levar a novas expansões
 - Entretanto, um comando frágil só funciona bem no ciclo simples expande-executa
 - ▶ Na prática, os comandos frágeis podem causar problemas quando passados como parâmetro

Exemplo prático: se um comando funciona bem no corpo do documento, mas não na legenda de uma figura, suspeito estar lidando com um comando frágil ²⁴

Para tornar um comando robusto, substitua o comando `\newcommand`. Use:

```
\DeclareRobustCommand{\nome}[num]{definição}
```

- As **caixas fantasmas** são úteis quando você quer inserir caixas no texto para acertar alinhamentos, mas não quer que o conteúdo delas apareça no resultado final. Use:

```
\phantom{ ... conteúdo fantasma ... }
```

²⁴Ver comandos frágeis em <http://www.forkosh.com/latex/ltx-2.html>

- Os comandos criados dentro de ambientes poderiam causar conflito no momento de acessar os parâmetros. Esse problema não ocorre porque:

- ▶ Os parâmetros do ambiente são acessados com **#n**, sendo n o número do parâmetro
- ▶ Os parâmetros do comando são acessados com **##n**, sendo n o número do parâmetro

```
% Criando o novo ambiente
\newenvironment{precos}{
  \newcommand{\pechincha}[2]{ \item ##1 $\rightarrow$ R\$ ##2 }
  \begin{itemize}
  }{
    \end{itemize}
}
```

Digite o exemplo e compile para ver o resultado

```
% Usando o novo ambiente
\begin{precos}
  \pechincha{Café}{1,00}
  \pechincha{Chá}{2,50}
\end{precos}
```

- As **variáveis** podem ser simuladas como comandos

```
\newcommand{\criavar}[1]{\newcommand{#1}{0}}  
\newcommand{\setavar}[2]{\renewcommand{#1}{#2}}  
  
%--- usando  
\criavar{\valor}  
\setavar{\valor}{35}  
valor = \valor
```

- Os **contadores** são muito úteis para numerar objetos, por exemplo, são usados nas listas

```
\newcounter{nomeDoContador}           % cria o contador  
  
\stepcounter{nomeDoContador}          % incrementa de 1  
\addtocounter{nomeDoContador}{num}    % incrementa de num  
  
\refstepcounter{nomeDoContador}       % retorna valor como texto {para label e ref.  
cruzada}  
\value{nomeDoContador}                % retorna valor como número (para cálculo)
```

- Pacote **calc** usado para cálculos básicos com contadores

```
\usepackage{calc}
% ...
\newcounter{mine}
\setcounter{mine}{2*17}
\themine % o estranho comando \the extrai o valor do contador
```

- Pacote **ifthen** usado para ter comando condicional

```
\usepackage{ifthen}
% ...
\ifthenelse{ \equal{\myvar}{true} }
{ Variável = true. }
{ Variável = false. }
```

- Pacote **xstring** usado para operações com strings

```
\usepackage{xstring}
% ...
\newcommand{\mystr}{Hello World!}
% ...
O string em \mystr tem \StrLen{\mystr}{} caracteres. \\
Pergunta: O string em \mystr contém a subpalavra 'Hello'? \\
Resposta: \IfSubStr{\mystr}{Hello}{true}{false}.
```

- **Pacote calculator** provê uma calculadora científica dentro do \LaTeX

```
\usepackage{calculator}  
% ...  
\MAX{3}{5}{\soluca}  
\SQUAREROOT{2}{\solucb}  
\ADD{\soluca}{\solucb}{\solucc}  
  
\soluca\ + \solucb\ = \solucc
```

Alguns comentários:

- Os nomes dos comandos providos são em maiúsculas
- Há vários implementados, incluindo funções científicas
- O resultado é retornado na “variável” criada, o último parâmetro
- Há várias constantes implementadas: `\numberPI`, `\numberE`, etc
- Existe um comando de atribuição: `\COPY{1.27}{\var}`
- Estão implementadas funções trigonométricas, exponenciais, etc
- Há comandos para fazer cálculo vetorial e com matrizes
- Praticamente tudo que uma calculadora avançada faz, este pacote provê

- **Pacote multido** usado para ter comando de repetição

Forma geral do comando:

```
\multido{variáveis}{repetições}{corpo}
```

Exemplo:

```
\usepackage{multido}
% ...
\multido{\i=0+1,\n=0+0.25}{8}{
  $x_{\i} = \n$ \\
}
```

Alguns comentários:

- Foram criadas duas variáveis, `\i` e `\n`
- `\i` inicia em 0 com incremento de 1
- `\n` inicia em 0 com incremento de 0.25
- o laço repete 8 vezes

- Pacote `pgffor` usado para ter comando de repetição

Forma geral do comando:

```
\foreach {variáveis} in {lista} {corpo}
```

Exemplos:

```
\usepackage{pgffor}
% ... exemplo A
% uma variável, loop simples
\foreach \i in {0, 4, 3, 5}{ \noindent $x_{\i} = \i$ \\\ }
% ... exemplo B
% uma variável, loop simples com ...
\foreach \i in {0, 1, ..., 8}{ \noindent $x_{\i} = \i$ \\\ }
% ... exemplo C
\foreach \i / \j / \k in {0/0/1, 1/0/2, 2/3/3.5}{ \noindent $x_{\i, \j} = \k$ \\\ }
```

Alguns comentários:

- O exemplo A é auto explicativo
- No exemplo B, a elipse indica uma repetição, o LaTeX calcula o incremento
- No exemplo C, há três variáveis (`\i`, `\j`, `\k`) separadas por `/`
- Elas assumem os valores indicados nas triplas $\mathbf{v}_1/\mathbf{v}_2/\mathbf{v}_3$
- A elipse não funciona com mais de uma variável
- O pacote define dois comandos: `\foreach` e `\breakforeach`

- Implementa macros para melhor tratamento de parâmetros, as principais são:

`\newcommandx` `\renewcommandx`
`\newenvironmentx` `\renewenvironmentx`

- Vamos examinar uma, as outras são análogas

```
\newcommandx{\comando}[num][padrões]{ ... definição ...}
```

Alguns comentários:

- A novidade está nos **[padrões]** definidos para os parâmetros opcionais
- Os parâmetros sem valores padrão são obrigatórios

```
\usepackage{xargs}
% ...
\newcommandx{\coord}[3][1=1, 3=n]{(#2_{#1}, \ldots, #2_{#3})}
```

```
$\coord{x}$
$\coord{0}{y}$
$\coord{z}{m}$
$\coord{0}{t}{m}$
```

$$(x_1, \dots, x_n)$$

$$(y_0, \dots, y_n)$$

$$(z_1, \dots, z_m)$$

$$(t_0, \dots, t_m)$$

- Existem alguns “truques” para evitar ambiguidade nos parâmetros opcionais
- A opção **usedefault** e o salto de parâmetros com **[]**

```
\usepackage{xargs}
% ...
\newcommandx{\coord}[3][2=1, 3=n, usedefault]{(#2_{#1}, \ldots, #2_{#3})}
```

```
$\coord{x}$
$\coord{y}[0]$
$\coord{z}[ ][m]$
$\coord{t}[0][m]$
```

$$(1_x, \dots, 1_n)$$

$$(0_y, \dots, 0_n)$$

$$(1_z, \dots, 1_m)$$

$$(0_t, \dots, 0_m)$$

- Redefinindo o critério para uso do padrão

```
\usepackage{xargs}
% ...
\newcommandx{\test}[2][1=A, 2=B, usedefault=@]{(#1, #2)}
```

```
$\test[b]$
$\test[ ][b]$
$\test[@][b]$
```

$$(b, B)$$

$$(\cdot, b)$$

$$(A, b)$$

- Implementa macros para melhor tratamento de parâmetros, as principais são:

<code>\NewDocumentCommand</code>	<code>\RenewDocumentCommand</code>
<code>\NewDocumentEnvironment</code>	<code>\RenewDocumentEnvironment</code>
<code>\IfNoValue (TF)</code>	<code>\IfBoolean (TF)</code>

- O pacote permite implementar construções estranhas e/ou experimentais. O relatório de erros no caso de mal uso ainda não está totalmente estável. Use com cautela
- Vamos examinar uma macro, as outras são análogas ou intuitivas

```
\NewDocumentCommand{\comando}{especificações}{ ... definição ... }
```

A grande novidade está nas **{especificações}**, uma lista contendo uma especificação por parâmetro do comando. Existem várias opções, seguem algumas:

Especificação	Descrição breve
m	parâmetro obrigatório padrão, entre chaves
o	parâmetro opcional padrão, entre colchetes
O{valor}	parâmetro opcional, entre colchetes, com <i>default</i>
g	parâmetro opcional padrão, entre chaves
G{valor}	parâmetro opcional, entre chaves, com <i>default</i>
s	asterisco opcional no fim do nome do comando, vai resultar <code>\BooleanTrue</code> se o asterisco estiver presente, ou <code>\BooleanFalse</code> caso contrário

■ Exemplos:

```
\usepackage{xparse}

\NewDocumentCommand{\foo}{\s m G{3} o }
{
  \noindent
  \IfBooleanTF{#1}{1 = True \\\}{1 = False \\\}
  2 = #2 \\\ 3 = #3 \\\ 4 = #4
}
```

```
\foo{teste 1}
\foo*{teste 2}
\foo{teste 3}{5}
\foo{teste 4}[6]
\foo{teste 5}{7}[8]
```

```
1 = False; 2 = teste1; 3 = 3; 4 = -NoValue-
1 = True; 2 = teste2; 3 = 3; 4 = -NoValue-
1 = False; 2 = teste3; 3 = 5; 4 = -NoValue-
1 = False; 2 = teste4; 3 = 3; 4 = 6
1 = False; 2 = teste5; 3 = 7; 4 = 8
```

- Implementa três comandos, mas vamos examinar somente o primeiro:
`\NewEnviron` `\Collect@Body` `\collect@body`
- O comando `\NewEnviron` é uma alternativa para criação de ambiente

```
%--- criação de ambiente na versão tradicional
\newenvironment{nome}[num]{ antes }{ depois }

%--- criação de ambiente na versão do pacote environ
\NewEnviron{nome}[num]{ ... definição ... }{ ... acabamento ... }
```

Alguns comentários para a versão do pacote:

- O código da definição é apresentado num único bloco, isso facilita a programação
- O conteúdo do autor, contido no ambiente, é coletado pela macro `\BODY`
- O `{acabamento}` é opcional e executado depois do `\end{ nome }`

■ Exemplos:

```

\environbodyname\bodymoldura
\NewEnviron{moldura}[4]{
  \noindent
  \begin{tikzpicture}[scale=1]
    % Local definitions
    \tikzstyle{texto}=[anchor=base,
      text width={#2}, % rounded corners,
      fill={#1},draw={#3},inner sep={#4}]

    % Body
    \draw (0,0) node [style=texto] {\bodymoldura} ;
  \end{tikzpicture}
} % fim moldura

```

Digite o conteúdo do exemplo e compile para ver o resultado

```

\begin{moldura}{green!10}{\textwidth}{white}{2mm}
  Este tipo de moldura foi utilizado na confecção dos slides deste curso.
  Combinado com os recursos para controle de parâmetros, tornando esses
  parâmetros opcionais com valores default, a usabilidade do ambiente fica
  excelente.
\end{moldura}

```

■ O comando `\environbodyname` personaliza a macro `\BODY`

- Implementa macros para tratamento de parâmetros com opções nomeadas do tipo:

```
\MeuComando[cor=red, tam=2cm, estilo=normal]{Exemplo usando parâmetros nomeados}
```

- Este pacote é bastante complexo e difícil de dominar nos detalhes, mas como o resultado é muito elegante vale o esforço para aprender. Aqui será apresentado apenas um exemplo prático básico. Faça alterações com cuidado e ao próprio risco ;)
- Aqui nos interessam as seguintes macros que serão utilizadas no exemplo:

<code>\define@key</code>	define uma opção nomeada associada ao parâmetro
<code>\setkeys</code>	define valores <i>default</i> e amarra as opções nomeadas ao comando

- Além destas macros do pacote, serão utilizados no exemplo três comandos do T_EX :

<code>\makeatletter</code>	marca início de contexto ¹ para definições de opções nomeadas
<code>\makeatother</code>	marca fim de contexto ¹ para definições de opções nomeadas
<code>\def</code>	define novo comando (não faz checagem de segurança)

¹ Trata o caractere @ como letra na formação dos *tokens*, mas não faça você isso

■ Exemplo

```
\usepackage{xparse, xkeyval}

% ===== define as keys para o 1o parametro
\makeatletter
\define@key{MeuComando}{cor}{\def\comandocor{#1}}
\define@key{MeuComando}{tam}{\def\comandotam{#1}}
\define@key{MeuComando}{estilo}{\def\comandoestilo{#1}}
\setkeys{MeuComando}{cor={black}, tam={2mm}, estilo={normal}}
\makeatother

% ===== define o comando
\DeclareDocumentCommand{\MeuComando}{o m}{
\begin{group}
\setkeys{MeuComando}{#1}
\noindent \hrulefill \par
Parâmetro obrigatório = #2 \par
Opções do parâmetro opcional: \par
\quad cor = \comandocor \par
\quad tam = \comandotam \par
\quad estilo = \comandoestilo \par
\noindent \hrulefill \par
\end{group} }
```

■ Digite o código do exemplo, compile e execute para ver o resultado

```
\MeuComando[] { teste 1 }
\MeuComando[cor=red] { teste 2 }
\MeuComando[estilo=poor, cor=green] { teste 3 }
\MeuComando[tam=2cm] { teste 4 }
```

11.7. Pacote etoolbox

- Implementa diversas macros e comandos para auxiliar na programação
- Implementa também uma série de “ganchos” (*hooks*) que não serão analisados aqui
- Exemplos do que encontrar no pacote:
 - Várias macros para tratar comandos frágeis
 - Vários comandos para tratamento de variáveis, contadores, booleanos etc
 - Várias variantes de comandos de decisão (*if's*)
 - Comandos aritméticos e lógicos, operadores, etc
 - Implementa os parênteses para uso em expressões
 - Comandos para tratamento de listas, incluindo comandos de repetição
 - E muito mais
- O pacote também define uma série de “ganchos” (*hooks*) que não serão analisados aqui
- Para mais detalhes, consulte a documentação do pacote

Criação de pacotes

12 Criação de pacotes

- Idéia geral

- Quando você define muitos comandos e ambientes, o preâmbulo começa a ficar muito longo. Talvez seja uma boa idéia criar um pacote para suas definições
- Pacotes e classes são semelhantes, mas os pacotes são mais flexíveis. Aqui vamos examinar apenas a criação de pacotes, para maior informação consulte os manuais
- Exemplo: pacote pacotinho, arquivo **pacotinho.sty**

```
%--- pacotinho.sty
\NeedsTeXFormat{LaTeX2e}[1994/06/01]
\ProvidesPackage{pacotinho}[2016/02/14 pacotinho Package]

%--- lista de pacotes que serão incluídos junto com o pacotinho
\RequirePackage[brazil]{babel}
\RequirePackage[T1]{fontenc}
\RequirePackage[latin1]{inputenc}

%--- trata opções, se houver

%--- comandos e ambientes exportados
\providecommand{\aspas}[1]{''#1''}

%--- fim do pacote
\endinput
```

- Para criar opções nomeadas para seu pacote, use:

<code>\DeclareOption{ opção }</code>	<code>{ ... ação ... }</code>	: um comando por opção
<code>\ExecuteOptions{ opção }</code>		: define opção padrão
<code>\ProcessOptions\relax</code>		: termina processamento das opções

- Você pode exportar um comando definindo-o com `\providecommand`
- Atenção, **não** existe um comando para exportar ambientes, por isso recomendo a criação de pacotes utilizando o pacote `xparse` que provê os comandos:

`\ProvideDocumentCommand`
`\ProvideDocumentEnvironment`

- Comandos e ambiente criados com os tradicionais `\newcommand` e `\newenvironment` são locais ao pacote e **não** serão exportados
- O último comando do pacote deve ser `\endinput`

Referências

13 Referências

13. Materiais Consultados (1/2)



CTAN

Comprehensive TeX Archive Network

www.ctan.org



TeKample.net

TikZ & PGF Manual

www.texample.net/media/pgf/builds/pgfmanualCVS2012-11-04.pdf



The L^AT_EX Project

L^AT_EX documentation

<https://latex-project.org/guides>



The L^AT_EX 3 Project

L^AT_EX 2 ϵ for class and package writers

www.lasca.ic.unicamp.br/pub/ctan/macros/latex/doc/clsguide.pdf



Wikibooks

LaTeX

<https://en.wikibooks.org/wiki/LaTeX>

13. Materiais Consultados (2/2)



KOTTWITZ, Stefan

LaTeX beginner's guide

Packt Publishing Ltd, 2011



KOTTWITZ, Stefan

LaTeX Cookbook

Packt Publishing Ltd, 2015



LAMPORT, Leslie

Latex: A Document Preparation System

Addison Wesley, 1994



MITTELBACH, F., GOOSSENS, M.

The LaTeX Companion

Addison Wesley, 2004



FLYNN, Peter

A Beginner's Introduction to Typesetting with LaTeX

Comprehensive TeX Archive Network, 2005

The End