Conditional Commands in LaTeX

M. Roos

April 19, 2025

1 Define parameter

- $i = 1$
- $j = 2$
- $k = -1$
- $x = 1.1cm$
- $y = 2pt$
- $z = 2.0pt$

2 Native commands

2.1 The \ifodd command

Description: Checks if a number <i> is odd or even.
Inline syntax:

```
\ifodd <i> <code if true> \else <code if false> \fi
```

Block syntax:

```
\ifodd <i>
<code executed if condition is true>
\else
<code executed if condition is false>
\fi
```

Command	Result
\ifodd \i	odd
\ifodd \j	even
\ifodd \k	odd
\ifodd \y	even
\ifodd \z	even

Example: `\ifodd \i {Odd} \else {Even} \fi` $\Longrightarrow$ produces "Odd ".

2.2 The \ifnum command

Description: Compares two integer numbers <i> and <j> and conditionally executes code.
Valid Op: < (less than), = (equal to), > (greater than)
Inline syntax:

```
\ifnum <i> <Op> <j> <code if true> \else <code if false> \fi
```

Block syntax:

```
\ifnum <i> <Op> <j>
    <code executed if condition is true>
\else
    <code executed if condition is false>
\fi
```

Op	Command	Result
<	\ifnum \i < \j	true
<	\ifnum \i < \k	false
>	\ifnum \i > \j	false
>	\ifnum \i > \k	true
=	\ifnum \i = \j	false
=	\ifnum \i = \k	false

Example:

```
\ifnum \i < \j
   True
\else
   False
\fi %produces "\ifnum \i<\j True \else False \fi"
```

2.3 The \ifdim command

Description: Compares two dimensions <x> and <y> and conditionally executes code.
Valid Op: < (less than), = (equal to), > (greater than)
Inline syntax:

```
\ifdim <x> <Op> <y> <code if true> \else <code if false> \fi
```

Block syntax:

```
\ifdim <x> <Op> <y>
   <code executed if condition is true>
\else
   <code executed if condition is false>
\fi
```

Op	Command	Result
<	\ifdim \x < \y	false
<	\ifdim \z < \y	false
>	\ifdim \x > \y	true
>	\ifdim \z > \y	false
=	\ifdim \x = \y	false
=	\ifdim \z = \y	true

Example: `\ifdim \x < \y True \else False \fi` $\Longrightarrow$ produces "False ".

2.4 The \ifcase command

Description: Compares a number <i> with a list of numbers and conditionally executes code. Implements a multiple selection structure, similar to the switch-case of other programming languages.
Inline syntax:

```
\ifcase <i> <cond 0> \or <cond 1> \or <cond 2> \else <cond n> \fi
```

Block syntax:

```
\ifcase <i>
   <code executed if <i> is 0>
\or
   <code executed if <i> is 1>
\or
   <code executed if <i> is 2>
\else
   <code executed if <i> is n> % Standart case
\fi
```

Command	Result
\ifcase \i	condition 1
\ifcase \j	condition 2
\ifcase \x	condition 2
\ifcase \y	condition 2

Example: `\ifcase \i {condition 0} \or {condition 1} \fi` $\Longrightarrow$ produces "condition 1 ".

2.5 The \ifx or \if command

Description: Compares two tokens <token 1> and <token 2> and conditionally executes code.
Inline syntax:

```
\ifx <token 1> <token 2> <code if true> \else <code if false> \fi
%or
\if <token 1> <token 2> <code if true> \else <code if false> \fi
```

Block syntax:

```
\ifx <token 1> <token 2>
   <code executed if token 1 is identical to token 2>
\else
   <code executed if condition is false>
\fi
%or
\if <token 1> <token 2>
   <code executed if token 1 is identical to token 2>
\else
   <code executed if condition is false>
\fi
```

Command	Result
\ifx \i \j	false
\ifx \i \k	false
\ifx \x \y	false
\ifx \z \y	false

Example: `\ifx \i \j {True} \else {False} \fi` $\Longrightarrow$ produces "False ".

2.6 The \newif command

Description: You can create new conditionals (as a kind of boolean variables) with the `\newif` command. With this self defined conditionals you can control the output of your code in an elegant way. Two versions of a document must be generated.

Inline syntax:

```
\newif\if<boolFlag> \boolFlagtrue \ifboolFlag <code if true> \else
   <code if false> \fi
```

Block syntax:

```
\newif\if<boolFlag> % create a new boolean variable
\boolFlagtrue       % set the boolean variable to true
%\boolFlagfalse     % set the boolean variable to false
\ifboolFlag         % check the value of the boolean variable
   <code executed if condition is true>
\else
   <code executed if condition is false>
\fi
```

Command	Result
\ifboolFlag	**True**

Example:

```
\newif\ifOp % create a new boolean variable
\Optrue     % set the boolean variable to true
\ifOp       % check the value of the boolean variable
   \textbf{True}
\else
   \textbf{False}
\fi
```

produces "<code executed if condition is false> ".

2.7 The \ifdefined command

Description: Checks if a command <Op> is defined or not.
Inline syntax:

```
\ifdefined <Op> <code if true> \else <code if false> \fi
```

Block syntax:

```
\ifdefined <Op>
   <code executed if condition is true>
\else
   <code executed if condition is false>
\fi
```

Command	Result
\ifdefined \i	defined
\ifdefined \j	defined
\ifdefined \k	defined
\ifdefined \abc	undefined

Example: \ifdefined \i {True} \else {False} \fi $\Longrightarrow$ produces "True ".

3 Packages

3.1 The \ifthen package

Description: Provides a more flexible and user-friendly way to handle conditional statements in LaTeX. It allows use combinational logic operators and, or and not. Indeed, the pair of parenthesis () is used to group the boolean expressions. However, the etoolbox package is recommended for new documents, because it is more modern and has a more consistent syntax. Indeed, the etoolbox package is developed with eLatex in mind, while the ifthen package is a legacy package.

3.1.1 The `\ifthenelse` command

Description: Compares an integer positive `<i>` and conditionally executes code.
Valid Op:

- Combinational logical operators: `and` and `or`;
- Negation operator: `not`;
- Relational logical operators: `<` (less than), `>` (greater than), `=` (equal to);
- `()` is used to group the boolean expressions;
- `\isodd` check if a number is odd or even. Similar to `\ifodd`;
- `\isundefined` is true if `\cmd` is not defined. Similar to `\ifdefined`;
- `\equal` check if two strings is equal or not. Similar to `\ifx`;
- `\lengthtest` check if a length is lesser, greater or equal to another length. Similar to `\ifdim`;
- `\boolean` check if a boolean variable is true or false. Similar to `\newif` mechanism.

Inline syntax:

```
\usepackage{ifthen}
\ifthenelse{<boolean expression>}{<code if true>}{<code if false>}
```

Block syntax:

```
\usepackage{ifthen}
\ifthenelse{<boolean expression>}
   {<code if true>
}{
   <code if false>}
```

Op	**Command**	**Result**
<	`\ifthenelse{\i < \j}`	true
<	`\ifthenelse{\i < \k}`	false (*Mathematical absurdity*)
<	`\ifthenelse{\i < \x}`	false (*Mathematical absurdity*)
<	`\ifthenelse{\i < \y}`	pttrue (*Undefined*)
=	`\ifthenelse{\i = \j}`	false
=	`\ifthenelse{\i = \k}`	false
=	`\ifthenelse{\y = \z}`	2pt=2.0pttruefalse (*Undefined*)
`\and`	`\ifthenelse{\i < \j \and 0 < \i}`	true
`\or`	`\ifthenelse{\i < \j \or \i > \j}`	true
`\not`	`\ifthenelse{\not \i < \j}`	false
`\AND`	`\ifthenelse{\i < \j \AND 0 < \i}`	true

\OR	\ifthenelse{\i < \j \OR \i > \j}	true
\NOT	\ifthenelse{\NOT \i < \j}	false
\(\)	\ifthenelse{\(\i < \j\)}	true
\isodd	\ifthenelse{\isodd{\i}}	true
\isodd	\ifthenelse{\isodd{\j}}	false
\isodd	\ifthenelse{\isodd{\k}}	true
\isodd	\ifthenelse{\isodd{\y}}	false
\isodd	\ifthenelse{\isodd{\z}}	false
\isundefined	\ifthenelse{\isundefined{\i}}	false
\isundefined	\ifthenelse{\isundefined{\j}}	false
\isundefined	\ifthenelse{\isundefined{\k}}	false
\isundefined	\ifthenelse{\isundefined{\abc}}	true
\equal	\ifthenelse{\equal{\i}{\i}}	true
\equal	\ifthenelse{\equal{\i}{1}}	true
\equal	\ifthenelse{\equal{\i}{\j}}	false
\equal	\ifthenelse{\equal{\i}{\k}}	false
\equal	\ifthenelse{\equal{\x}{\y}}	false
\equal	\ifthenelse{\equal{\z}{\y}}	false
\lengthtest and <	\ifthenelse{\lengthtest{\x < \y}}	false
\lengthtest and <	\ifthenelse{\lengthtest{\z < \y}}	false
\lengthtest and >	\ifthenelse{\lengthtest{\x > \y}}	true
\lengthtest and >	\ifthenelse{\lengthtest{\z > \y}}	false
\lengthtest and =	\ifthenelse{\lengthtest{\x = \y}}	false
\lengthtest and =	\ifthenelse{\lengthtest{\z = \y}}	true
\boolean	\ifthenelse{\boolean{boolFlag}}	true

Example: `\ifthenelse{\i<\j}{True}{False}` $\Longrightarrow$ produces "True".
Extra:

- `\newboolean`: Creates a new boolean variable. If it the variable already exists, an error is thrown. Ex: `\newboolean{boolFlag}`;
- `\provideboolean`: Creates a new boolean variable, even if it already exists.

  Ex: `\provideboolean{boolFlag}`;
- `\setboolean`: Sets the value of a boolean variable. Ex: `\setboolean{boolFlag}{true}` or `\setboolean{boolFlag}{false}`;

3.2 The \etoolbox package

Description: Provides a more flexible and user-friendly way to handle conditional statements in LaTeX. This package provides two interfaces to boolean flags which are completely independent of each other. The tools in section 3.5.1 are a LaTeX frontend to `\newif`. Those in section 3.5.2 use a different mechanism.
Inline syntax:

```
\usepackage{etoolbox}
\ifboolexpr{<condition>}{<code if true>}{<code if false>}
```

Block syntax:

```
\usepackage{etoolbox}
\ifboolexpr{<condition>}
        {<code if true>
}{
        <code if false>}
```

Command	Result
\ifboolexpr{\i<\j}	

3.3 The \xstring package