Abstract

This package allows typesetting pseudocode in \LaTeX. It is based on algpseudocode from the algorithmicx package and uses the same syntax, but adds several new features and improvements. Notable features include customizable indent guide lines and the ability to draw boxes around parts of the code for highlighting differences. This package also has better support for long code lines spanning several lines and improved comments.
1 Basic Usage

To use the package, load it in your preamble:

\usepackage{algpseudocodex}

Basic usage is identical to \algseudocode from the algorithmicx package. Pseudocode written for that package should also be compatible with algpseudocodex.

1.1 Algorithmic Block

Pseudocode can be typeset inside a algorithmic blocks:

\begin{algorithmic}[line numbering]
...
\end{algorithmic}

The optional argument specifies how lines are numbered. 0 means no numbering, \( n > 0 \) means every \( n \)th line gets a number. The default is 0, i.e., no line numbers will be typeset if no optional argument is provided.

1.2 Simple Statements and Commands

Statements start with \texttt{State}. The command \texttt{Statex} can be used to start a new line that does not get a new line number.

The commands \texttt{Return} and \texttt{Output} can be used for return values of functions and outputs. They do not start a new line on their own, so they need to be used together with \texttt{State}.

The \texttt{Call} command is used for procedure calls. It takes two arguments: The first one is the name of the procedure and the second one are the arguments.

Example

\begin{algorithmic}[1]
  \State first line
  \Statex continuing first line
  \State \Call{Proc}{a1, a2}
  \State \Output Hello World!
\end{algorithmic}

1.3 Blocks

Blocks are used for loops, conditional statements, and functions. Blocks can also be nested within other blocks.

1.3.1 While Loop

\begin{algorithmic}
  \While{condition}
    \State body
  \EndWhile
\end{algorithmic}

\hspace{1cm} \textbf{while condition do}
\hspace{1cm} \begin{array}{l}
  \text{body}
\end{array}

1.3.2 For Loop

\begin{algorithmic}
  \For{$n = 1, \ldots, 10$}
    \State body
  \EndFor
\end{algorithmic}

\hspace{1cm} \textbf{for } n = 1, \ldots, 10 \text{ do}
\hspace{1cm} \begin{array}{l}
  \text{body}
\end{array}
1.3.3 For-All Loop

\ForAll{$n \in \{1, \ldots, 10\}$}
\State body
\EndFor

for all $n \in \{1, \ldots, 10\}$ do
\_ body

1.3.4 Loop

\Loop
\State body
\EndLoop

loop
\_ body

1.3.5 Repeat-Until Loop

\Repeat
\State body
\Until{$n > 10$}

repeat
\_ body
until $n > 10$

1.3.6 If Statement

\If{condition}
\State body
\ElsIf{condition}
\State body
\Else
\State body
\EndIf

if condition then
\_ body
else if condition then
\_ body
else
\_ body

The \ElsIf and \Else parts are optional.

1.3.7 Procedure

\Procedure{name}{parameters}
\State body
\EndProcedure

procedure NAME(parameters)
\_ body

1.3.8 Function

\Function{name}{parameters}
\State body
\EndFunction

function NAME(parameters)
\_ body

1.4 Require and Ensure

To specify conditions on the inputs and outputs of an algorithm, \Require and \Ensure can be used.

Example

\begin{algorithmic}[1]
\Require $x \in \{0,1\}$
\Ensure $y \in \{1,2\}$
\State $y \gets x+1$
\State \Return $y$
\end{algorithmic}

Require: $x \in \{0, 1\}$
Ensure: $y \in \{1, 2\}$
1: $y \leftarrow x + 1$
2: return $y$
1.5 Comments

There are two ways to typeset code comments: The command `\Comment` can be used to add short comments to the end of the current line. The command `\LComment` can be used to typeset long comments that can span multiple lines. Comments with `\LComment` start on a new line.

Example

```
\begin{algorithmic}[1]
    \State $x \gets y^2$
    \LComment{The next two lines increment both $x$ and $y$.}
    \State $x \gets x + 1$
    \Comment{Increment $x$.}
    \State $y \gets y + 1$
    \Comment{Increment $y$.}
\end{algorithmic}
```

2 Boxes

A unique feature of the `algpseudocodex` package is the ability to draw boxes around pieces of code. There are two different methods to do so: One for drawing boxes around multiple lines of code, and another one for drawing a box around a string on a single line of code.

2.1 Boxes Around Multiple Lines of Code

The command `\BeginBox[style]` is used to set the beginning of the box. The optional argument determines the style of the drawn box. The boxes are drawn using TikZ, so any TikZ style can be used. The default style can be changed as described in Section 4.2. The command `\EndBox` is used to set the end of the last started box. Boxes can be nested arbitrarily, but every `\BeginBox` needs a matching `\EndBox`.

Example

```
\begin{algorithmic}
    \BeginBox
        \State first line
    \BeginBox[fill=yellow]
        \State second line
        \State another line
    \EndBox
    \EndBox
    \BeginBox[draw=blue,dashed]
        \State last line
    \EndBox
\end{algorithmic}
```

2.2 Boxes Inside Single Line

The command `\BoxedString[style]{text}` is used to typeset text with a box around it. The optional argument determines the style of the box, as in `\BeginBox`. The default style is the same as for `\BeginBox`.
Example
\begin{algorithmic}
\State first line
\State second line with \texttt{\textcolor{red}{box}}
\State last line
\end{algorithmic}

3 Package Options

When loading \texttt{algpseudocodex} the options describe in this section can be set. They syntax for setting \texttt{option1=\textcolor{red}{value1}} and \texttt{option2=\textcolor{red}{value2}} is:

\texttt{\usepackage[option1=value1,option2=value2]{algpseudocodex}}

3.1 noEnd

possible values: \texttt{true, false}

default: \texttt{true}

If \texttt{false}, the end of blocks are marked with the expression “end” followed by the name of the block.

Example
\begin{verbatim}
noEnd=false:
  if \textcolor{red}{x > 0} then
    \texttt{x ← x − 1}
  end if

noEnd=true:
  if \textcolor{red}{x > 0} then
    \texttt{x ← x − 1}
\end{verbatim}

3.2 indLines

possible values: \texttt{true, false}

default: \texttt{true}

If \texttt{true}, indent guide lines are drawn. The style of the lines can be customized as described in Section 4.1.

Example
\begin{verbatim}
indLines=false:
  if \textcolor{red}{x > 0} then
    \texttt{x ← x − 1}

indLines=true:
  if \textcolor{red}{x > 0} then
    \texttt{x ← x − 1}
\end{verbatim}

3.3 spaceRequire

possible values: \texttt{true, false}

default: \texttt{true}

If \texttt{true}, vertical space is added before every \texttt{\textcolor{red}{\text{Require}}} except the one on the first line. This is useful for specifying different behaviors depending on the provided input.
Example

\text{spaceRequire=false:}
\begin{align*}
\text{Require: } x &\in \{0, 1\} \\
& \quad \text{return } x \\
\text{Require: } x &\in \{1, 2\} \\
& \quad \text{return } x - 1
\end{align*}
\text{spaceRequire=true:}
\begin{align*}
\text{Require: } x &\in \{0, 1\} \\
& \quad \text{return } x \\
\text{Require: } x &\in \{1, 2\} \\
& \quad \text{return } x - 1
\end{align*}

3.4 italicComments

\text{possible values: true, false}
\text{default: true}

If \text{true}, all comments are typeset in italic font. If \text{false}, comments are typeset in roman font.

Example

\text{italicComments=false:}
\begin{align*}
& \quad \triangleright \text{Long comment.} \\
& \quad x \leftarrow 0 \\
& \quad x \leftarrow x^2 \triangleright \text{Does not fit on the current line and is thus not justified.}
\end{align*}
\text{italicComments=true:}
\begin{align*}
& \quad \triangleright \text{Long comment.} \\
& \quad x \leftarrow 0 \\
& \quad x \leftarrow x^2 \triangleright \text{Does not fit on the current line and is thus not justified.}
\end{align*}

3.5 rightComments

\text{possible values: true, false}
\text{default: true}

If \text{true}, comments typeset with \texttt{\Comment} are right justified on the current line. If a comment does not fit on the current line, no justification is applied. If \text{false}, all comments are typeset right after the end of the current line.

\text{Does not affect long comments typeset with \texttt{\LComment}.}

Example

\text{rightComments=false:}
\begin{align*}
& \quad \triangleright \text{No effect on long comments.} \\
& \quad x \leftarrow 0 \triangleright \text{Short comment.} \\
& \quad x \leftarrow x^2 \triangleright \text{Does not fit on the current line and is thus not justified.}
\end{align*}
\text{rightComments=true:}
\begin{align*}
& \quad \triangleright \text{No effect on long comments.} \\
& \quad x \leftarrow 0 \\
& \quad x \leftarrow x^2 \triangleright \text{Does not fit on the current line and is thus not justified.}
\end{align*}

3.6 commentColor

\text{possible values: Any color that can be used in \texttt{\textcolor{color}{text}}.}
\text{default: gray}

Defines the color in which comments are typeset.

Example

\text{commentColor=black:}
\begin{align*}
& \quad \triangleright \text{Long comment.} \\
& \quad x \leftarrow 0 \\
& \quad x \leftarrow x^2 \triangleright \text{Short comment.}
\end{align*}
\text{commentColor=blue:}
\begin{align*}
& \quad \triangleright \text{Long comment.} \\
& \quad x \leftarrow 0 \\
& \quad x \leftarrow x^2 \triangleright \text{Short comment.}
\end{align*}
3.7 beginComment and endComment

possible values: Any string that can be typeset in text mode.

default: $\triangleright$~ and (empty)

Used to indicate the beginning and end of comments typeset with \Comment, respectively.

Example

\beginComment=//~: \beginComment=/*~/:
\begin{algorithmic}
  \Comment Long comment. \Comment Long comment.
  x ← 0 \Comment// Short comment. x ← 0 \Comment/* Short comment. */
\end{algorithmic}

3.8 beginLComment and endLComment

possible values: Any string that can be typeset in text mode.

default: $\triangleright$~ and $\triangleleft$

Used to indicate the beginning and end of long comments typeset with \LComment, respectively.

Example

\beginLComment=/*~, endLComment=~/~:
\begin{algorithmic}
  /* Long comment. */
  x ← 0 \Comment// Short comment.
\end{algorithmic}

4 Customization

4.1 Style of Indent Guide Lines

Indent guide lines are drawn using TikZ and consequently any TikZ style can be used. To set the style, use:

\tikzset{algpxIndentLine/.style={style}}

The default style is draw=gray, very thin.

Example

\algpxIndentLine/.style={draw=blue,dashed}:
\begin{algorithmic}
  \algpxIndentLine if x > 0 then
  ↓ \algpxIndentLine x ← x - 1
\end{algorithmic}

4.2 Default Style of Boxes

Boxes are drawn using TikZ and consequently any TikZ style can be used. To set the default style, use:

\tikzset{algpxDefaultBox/.style={style}}

The default style is draw.
4.3 Changing Keywords

As in the algorithmicx package, keywords can be renamed using the syntax:

\algnewcommand\keyword{new name}

The following keywords can be customized:

- \algorithmicend  
  Default: \textbf{end}
- \algorithmicdo  
  Default: \textbf{do}
- \algorithmicwhile  
  Default: \textbf{while}
- \algorithmicfor  
  Default: \textbf{for}
- \algorithmicforall  
  Default: \textbf{for all}
- \algorithmicloop  
  Default: \textbf{loop}
- \algorithmicrepeat  
  Default: \textbf{repeat}
- \algorithmicuntil  
  Default: \textbf{until}
- \algorithmicprocedure  
  Default: \textbf{procedure}
- \algorithmicfunction  
  Default: \textbf{function}
- \algorithmicif  
  Default: \textbf{if}
- \algorithmicthen  
  Default: \textbf{then}
- \algorithmicelse  
  Default: \textbf{else}
- \algorithmicrequire  
  Default: \textbf{Require:}
- \algorithmicensure  
  Default: \textbf{Ensure:}
- \algorithmicreturn  
  Default: \textbf{return}
- \algorithmicoutput  
  Default: \textbf{output}

5 Revision History

v1.0.2 (2022-10-07)
- Fixed bug with incorrectly ended indent block for nested statements.

v1.0.1 (2021-12-05)
- Fixed bug regarding alignment of comments after end if, end for etc.

v1.0 (2020-08-16)
- Initial release.