The `algorithmicx` package*

Szász János
szaszjanos@users.sourceforge.net

2024 年 10 月 4 日

概要

The `algorithmicx` package provides many possibilities to customize the layout of algorithms. You can use one of the predefined layouts (**pseudocode**, **pascal** and **c** and others), with or without modifications, or you can define a completely new layout for your specific needs.

目次

1 Introduction

All this has begun in my last year at the university. The only thing that I knew of LaTeX was that it exists, and that it is "good". I started using it, but I needed to typeset some algorithms. So I begun searching for a good algorithmic style, and I have found the `algorithmic` package. It was a great joy for me, and I started to use it... Well... Everything went nice, until I needed some block that wasn't defined in there. What to do? I was no LaTeX guru, in fact I only knew the few basic macros. But there was no other way, so I opened the style file, and I copied one existing block, renamed a few things, and

* This is the documentation for the version 1.2 of the package. This package is released under LPPL.

voilà! This (and some other small changes) where enough for me...

One year later — for one good soul — I had to make some really big changes on the style. And there on a sunny day came the idea. What if I would write some macros to let others create blocks automatically? And so I did! Since then the style was completely rewritten... several times...

I had fun writing it, may you have fun using it! I am still no LaTeX guru, so if you are, and you find something really ugly in the style, please mail me! All ideas for improvements are welcome!

Thanks go to Benedek Zsuzsa, Ionescu Clara, Szőcs Zoltán, Cseke Botond, Kanoc and many-many others. Without them I would have never started or continued **algorithmicx**.

2 General informations

2.1 The package

The package **algorithmicx** itself doesn't define any algorithmic commands, but gives a set of macros to define such a command set. You may use only **algorithmicx**, and define the commands yourself, or you may use one of the predefined command sets.

These predefined command sets (layouts) are:

algpseudocode has the same look[*1] as the one defined in the `algorithmic` package. The main difference is that while the `algorithmic` package doesn't allow you to modify predefined structures, or to create new ones, the `algorithmicx` package gives you full control over the definitions (ok, there are some limitations — you can not send mail with a, say, `\For` command).

algcompatible is fully compatible with the `algorithmic` package, it should be used only in old documents.

algpascal aims to create a formatted pascal program, it performs automatic indentation (!), so you can transform a pascal program into an **algpascal** algorithm description with some basic substitution rules.

algc – yeah, just like the **algpascal**... but for c... This layout is incomplete.

To create floating algorithms you will need `algorithm.sty`. This file may or may not be included in the `algorithmicx` package. You can find it on CTAN, in the `algorithmic` package.

2.2 The algorithmic block

Each algorithm begins with the `\begin{algorithmic}[lines]` command, the optional `lines` controls the line numbering: 0 means no line numbering, 1 means number every line, and n means number lines n, $2n$, $3n$... until the `\end{algorithmic}` command, witch ends the algorithm.

[*1] almost :-)

2.3 Simple lines

A simple line of text is beginned with `\State`. This macro marks the begin of every line. You don't need to use `\State` before a command defined in the package, since these commands use automatically a new line.

To obtain a line that is not numbered, and not counted when counting the lines for line numbering (in case you choose to number lines), use the `Statex` macro. This macro jumps into a new line, the line gets no number, and any label will point to the previous numbered line.

We will call *statament*s the lines starting with `\State`. The `\Statex` lines are not statements.

2.4 Placing comments in sources

Comments may be placed everywhere in the source using the `\Comment` macro (there are no limitations like those in the `algorithmic` package), feel the freedom! If you would like to change the form in witch comments are displayed, just change the `\algorithmiccomment` macro:

```
\algrenewcommand{\algorithmiccomment}[1]{\hskip3em$\rightarrow$ #1}
```

will result:

1: $x \leftarrow x + 1$ → Here is the new comment

2.5 Labels and references

Use the `\label` macro, as usual to label a line. When you use `\ref` to reference the line, the `\ref` will be subtitued with the corresponding line number. When using the **algorithmicx** package togedher with the **algorithm** package, then you can label both the algorithm and the line, and use the `\algref` macro to reference a given line from a given algorithm:

```
\algref{<algorithm>}{<line>}
```

```
The \textbf{while} in algorithm
\ref{euclid} ends in line
\ref{euclidendwhile}, so
\algref{euclid}{euclidendwhile}
is the line we seek.
```

The **while** in algorithm 1 ends in line 7, so 1.7 is the line we seek.

2.6 Breaking up long algorithms

Sometimes you have a long algorithm that needs to be broken into parts, each on a separate float. For this you can use the following:

`\algstore{<savename>}` saves the line number, indentation, open blocks of the current algorithm and closes all blocks. If used, then this must be the last command before closing the algorithmic

block. Each saved algorithm must be continued later in the document.

`\algstore*{<savename>}` Like the above, but the algorithm must not be continued.

`\algrestore{<savename>}` restores the state of the algorithm saved under `<savename>` in this algorithmic block. If used, then this must be the first command in an algorithmic block. A save is deleted while restoring.

`\algrestore*{<savename>}` Like the above, but the save will not be deleted, so it can be restored again.

See example in the **Examples** section.

2.7 Multiple layouts in the same document

You can load multiple algorithmicx layouts in the same document. You can switch between the layouts using the `\alglanguage{<layoutname>}` command. After this command all new algorithmic environments will use the given layout until the layout is changed again.

3 The predefined layouts

3.1 The **algpseudocode** layout

If you are familiar with the `algorithmic` package, then you'll find it easy to switch. You can use the old algorithms with the **algcompatible** layout, but please use the **algpseudocode** layout for new algorithms.

To use **algpseudocode**, simply load `algpseudocode.sty`:

```
\usepackage{algpseudocode}
```

You don't need to manually load the **algorithmicx** package, as this is done by **algpseudocode**.

The first algorithm one should write is the first algorithm ever (ok, an improved version), *Euclid's algorithm*:

Algorithm 1 Euclid's algorithm

```
1: procedure EUCLID(a, b)                    ▷ The g.c.d. of a and b
2:     r ← a mod b
3:     while r ≠ 0 do                        ▷ We have the answer if r is 0
4:         a ← b
5:         b ← r
6:         r ← a mod b
7:     end while
8:     return b                              ▷ The gcd is b
9: end procedure
```