The $\Pi\rho\acute{o}\gamma\rho\alpha\mu\mu\alpha$ Package
version 0.5

Paulo G. S. da Fonseca
Federal University of Pernambuco–Brazil

May 31, 2005

Abstract

This is the documentation of the LaTeX package `programma`, which provides an environment for the typesetting of algorithms in pseudo-language from a simple set of macros which reflect the structure of the code. The package also provides a float environment for such algorithms, and a command for the generation of a List of Algorithms.

List of Algorithms

6.1 Example of the float `pgrm` environment. 5

1 Introduction

The LaTeX package `programma` provides an environment for the typesetting of algorithms in pseudo-language. From the outside, it seems like an enhanced(?) mutation of the `algorithm[mic]` package by Peter Williams, which is perhaps more elegantly written and certainly has a better manual :-). Though I've borrowed much of its syntax, this package has been written from scratch.

This package has been primarily written to be used along with the UFPEThesis LaTeX class (`www.cin.ufpe.br/~paguso/ufpethesis`). Nevertheless, `programma` has no dependency with respect to that class whatsoever, and thus can be independetly used by anyone.

I believe that, very often, one example is worth a dozen pages of explanation. This is pretty much the spirit of this manual!

2 All-in-one Example

Compare this output against the "source code" that follows.

```
Algorithm Test
Input     a: description of input a
          b: description of input b
Output    description of the output
  1 Simple statement
```

```
2  Statement {*in-line comments*}
3  If a long statement span various lines, like this one, it is still counted as a single
   line
4  {*Conditionals*}
5  **if** condition **then**
6      statement
7  **else if** condition **then**
8      statement
9  **else**
10     statement
11 **end if**
12 **switch** expression
13     **case** value
14         statement
15     **otherwise**
16         statement
17 **end switch**
18 {*Loops*}
19 **while** condition **do**
20     statement
21 **end while**
22 **do**
23     statement
24 **while** condition
25 **repeat**
26     statement
27 **until** condition
28 **for** $i$ **from** $\alpha$ **to** [**down to** ] $\omega$ **do**
29     statement
30 **end for**
31 **for each** $v$ **in** $V$ **do**
32     statement
33 **end for**
34 {*Return*}
35 **return** value
36 {*Constructs can be arbitrarily nested*}
37 **if** $\omega \geq \alpha$ **then**
38     Notice the cross-reference in the next line
39     **for** $i \leftarrow \alpha \dots \omega$ (I prefer this form over that of line 28) **do**
40         **switch** $i \mod 5$
41             **case** 0
42                 $j \leftarrow 0$
43                 **repeat**
44                     $j \leftarrow j + \log i$
45                 **until** $j > i$
46             **case** 1
47                 satement
48             **otherwise**
49                 statement
50         **end switch**
```

```
51      end for
52   end if
end
```

Here’s the source that produced the output above

```
\begin{programma}
\ALGORITHM{Test}
\INPUT
$a$: description of input $a$\\
$b$: description of input $b$
\ENDINPUT
\OUTPUT
description of the output
\ENDOUTPUT
  \STATE Simple statement
  \STATE Statement \COMMENT{in-line comments}
  \STATE If a long statement span various lines,
         like this one, it is still counted as
         a single line
  \STATE \COMMENT{Conditionals}
  \IF{condition}
    \STATE statement
  \ELSEIF{condition}
    \STATE statement
  \ELSE
    \STATE statement
  \ENDIF
  \SWITCH{expression}
    \CASE{value}
      \STATE statement
    \ENDCASE
    \OTHERWISE
      \STATE statement
    \ENDOTHERWISE
  \ENDSWITCH
  \STATE \COMMENT{Loops}
  \WHILE{condition}
    \STATE statement
  \ENDWHILE
  \DOWHILE{condition}
    \STATE statement
  \ENDDO
  \REPEAT
    \STATE statement
  \UNTIL{condition}
  \FOR{$i$ \FROM $\alpha$ \TO[\DOWNTO] $\omega$}\PGlnlabel{forline}
    \STATE statement
  \ENDFOR
  \FOREACH{$v$ \IN $V$}
    \STATE statement
  \ENDFOR
  \STATE\COMMENT{Return}
  \STATE\RETURN value
  \STATE\COMMENT{Constructs can be arbitrarily nested}
  \IF{$\omega\geq\alpha$}
    \STATE Notice the cross-reference in the next line
    \FOR{$i\GETS \alpha\ldots \omega$ (I prefer this form
                    over that of line~\PGlnref{forline})}
      \SWITCH{$i\mod 5$}
        \CASE{0}
          \STATE $j\GETS 0$
          \REPEAT
            \STATE $j\GETS j+\log{i}$
          \UNTIL $j>i$
        \ENDCASE
        \CASE{1}
          \STATE satement
        \ENDCASE
        \OTHERWISE
          \STATE statement
        \ENDOTHERWISE
      \ENDSWITCH
    \ENDFOR
  \ENDIF
\ENDALGORITHM
\end{programma}
```

3 Line numbers

Algorithms can come out with or without line numbers. By default, line numbers are shown. If you want your code without them, pass the argument `[1]` to the `programma` environment, that is, write

```
\begin{programma}[1]
```

4 END's

By default, the closing **end**'s of each block are not shown since, for my personal taste, they take up too much space for no big gain (indentation says it all). Anyway, if you want them in, then you must pass the option `showend` to the package, that is, write

```
\usepackage[showend]{programma}
```

in the preamble of your document[1].

[1]For the sake of consistency in the style, I've decided to set this option as global instead of allowing the user to choose it in a *per* algorithm basis. Nevertheless, it it still possible (though not adviceable) to control this behaviour explicitly through the commands `\showendtrue` and `\showendfalse` which turn line numbering on and off respectively.

```
Algorithm MSort
Input   $V = (v_1, \ldots, v_n)$: an assorted array of items
Output  The sorted array $V$
1  if $n = 1$ then
2      return $V$
3  else
4      return Merge( MSort( $(v_1, \ldots, v_{n/2})$ ), MSort( $(v_{n/2+1}, \ldots, v_n)$ ) )
5  end if
end
```

Algorithm 6.1: Example of the float `pgrm` environment.

5 Languages

The package `program` offers support to two languages, namely English and Portuguese. The support is achieved through the package options `en`, for English, the default language, and `pt`, for Portuguese. Thus, loading the package as in

`\usepackage[pt]{programma}`

will cause the keywords to show up in Portuguese. The package can also be easily adapted to support other languages.

6 Floats

The `programma` package also provides a floating environment named `pgrm`. For example, Algorithm 6.1 is produced by something like

```
\begin{pgrm}
\begin{programma}
\ALGORITHM{MSort}
...
\ENDALGORITHM
\end{programma}
\caption{Example of the float \texttt{pgrm} environment.}\label{alg:ms}
\end{pgrm}
```

By default, algorithms are numbered independently. If you want algorithms to be numbered within parts, chapters, sections, subsections or subsubsections, then you must pass the corresponding option, `part`, `chapter`, `section`, `subsection` or `subsubsection`, to the package.

A List of Algorithms, like the one in the beginning of this document can be produced via the macro `\listofalgorithms`.

7 Customisation

No documentation for the customisation of the package so far (and probably it will never exist). Nevertheless, the file is only sufficiently commented as to encourage hacking. Good luck!