ALgo

Writing algorithms in LaTeX.

Arnaud Lefebvre

ABISS – Université de Rouen – FRANCE
Arnaud.Lefebvre@univ-rouen.fr

Description

The ALgo extension is an easy to use set of commands including numerous options like line numbering, vertical rules for instructions blocks, different languages (you can define your own keywords and styles)...

Contents

---

1 First algorithm

The ALgo extension provides an environment simply called *algo*. This environment takes two arguments (the name of the algorithm and its parameters) and a series of options:

```
\begin{algo}[option1,option2...]{Name}{Parameters}
 instructions
\end{algo}
```

The easiest way to use it, without any option, is shown Fig.1.

As you can see, it is very simple. Let us now describe all the different commands available in this environment.

2 All the commands

Here are all the commands available in the ALgo package. For each command, an example is given.

2.1 Preliminary commands

The following commands have to be called just after the `\begin{algo}...` line and before the first instruction of the algorithm.

- `\IN{inputs}`: with this command you can specify the inputs of the algorithm:

  ```
  \IN{$i$, $j$ integers strictly greater than 0}
  ```

- `\OUT{outputs}`: with this command you can specify the outputs of the algorithm:

  ```
  \OUT{$m$ equal to $i\times j$}
  ```

- `\AUX{auxiliaries}`: with this command you can specify the other variables used in the algorithm:

  ```
  \AUX{$k$ integer}
  ```

```
\begin{algo}{Double}{i}
\RETURN{2\times i}
\end{algo}
```

DOUBLE(i)
1 **return** $2 \times i$

Figure 1: First algorithm. On the left, the source code. On the right, the corresponding algorithm.

Remark: if one of these three commands is used, keywords `BEGIN` and `END` are automatically added at the beginning and at the end of the algorithm.

2.2 Simple commands

- `\SET{i}{j}`: produces $i \leftarrow j$.
- `\INCR{i}`: produces $i \leftarrow i + 1$.
- `\DECR{i}`: produces $i \leftarrow i - 1$.
- `\CALL{Name}{Parameters}`: produces a call to another algorithm such as NAME(*Parameters*). This command can be used outside an algorithm.
- `\COM{comments}`: introduces a comment in your algorithm. This command can also be used as a parameter of another command. For example: `\SET{i}{\COM{two times $i$}}` will produce $i \leftarrow$ *two times* i. If this command is alone on a line, by default this line has no number and the comment starts with a ▷. If you want this line to be numbered, use the `numcom` option.
- `\ACT{value}`: writes the `value` in math mode;
- `\CUT`: enables to cut a long line (comment or loop condition for example). If you call this command, the end of the current line is put on the next line (with appropriate tabulations). This is very useful for long inputs.
- `\RETURN{value}`: produces RETURN *value*.
- `\BREAK`: produces BREAK.
- `\LABEL{label}`: introduces a label in the algorithm. This command has to be placed just after the instruction you want to label. A call to the famous `\ref` command will give the number of the line where the instruction appears.

2.3 Loops and conditions

This is the list of all the possible conditions and loops. The best way to understand each one is to have a look at Fig.2.

- `\IF{condition}...\FI` or `\IF{condition}...\ELSE...\FI`: just a simple condition instruction.
- `\IF{condition_1}...\ELSEIF{condition_2}...\FI` or `\IF{condition_1}...\ELSEIF{condition_2}...\ELSE...\FI`: multiple conditions instruction with or without a final `\ELSE` default case.
- `\DOWHILE{condition}...\OD`: for while loops.

- `\DO...\WHILEOD{condition}`: for while loops where the condition is verified at the end of the loop.
- `\DOFOR{sentence}...\OD`: useful for a loop where the bounds are not clearly defined. For example:

```
\DOFOR{each~line~of~the~file~F}...\OD
```

- `\DOFOREACH{sentence}...\OD`: no need to say more...
- `\DOFORI{var}{begin}{end}...\OD`: `var` takes all the values from `begin` upto `end`.
- `\DOFORD{var}{begin}{end}...\OD`: this time `var` takes all the values from `begin` downto `end`.
- `\DOFORIS{var}{begin}{end}{step}...\OD`: similar to `\DOFORI{var}{begin}{end}...\OD` but with a `step` between to values.
- `\DOFORDS{var}{begin}{end}{step}...\OD`: similar to `\DOFORD{var}{begin}{end}...\OD` but with a `step` between to values.
- `\REPEAT...\UNTIL{condition}`: no need to say more.

3 All the options

All the options presented in this section can be given through the

```
\usepackage[option1,option2...]{ALgo}
```

command (in this case these options are given for all the algorithms in the document) or through the

```
\begin{algo}[option1,option2...]
```

environment (in this case, these options are given for the current algorithm only).

Here are all the options available in the ALgo package:

- `noeqtab`: if you want tabulation sizes depending on the loop you are in.
- `ends`: if you want to add ends of loops.
- `rules`: if you want to draw vertical rules delimiting loops.
- `nonum`: if you don't want the numbers of the lines.
- `numcom`: if you want comment lines to be numbered. Of course this option is taken into account if the `nonum` option is not used.

```
\begin{algo}{Example}{i}
\IN{$i$ positive integer}
\OUT{$j$ equal to $i$}
\AUX{$k$ integer}
\IF{i=1}
  \SET{j}{i}
\ELSEIF{\COM{OK}}
  \SET{j}{0}
\ELSE
  \SET{j}{0}
\FI
\SET{k}{0}}
\REPEAT
  \INCR{j}
\UNTIL{j>i}
\DOFORI{k}{0}{i}
  \SET{j}{k}
\OD
\DOFORDS{k}{j}{0}{1}
  \SET{i}{\ACT{i}}
\OD
\DOWHILE{k<i}
 \SET{j}{\CALL{Double}{k}}
 \INCR{k}
\OD
\COM{Now $j=i$}
\SET{j}{(j/2)+1}
\RETURN{j}
```

EXAMPLE(i)
▷ **Input:** i positive integer
▷ **Output:** j equal to i
▷ **Auxiliary:** k integer
1 **Begin**
2 **if** $i = 1$ **then**
3 $j \leftarrow i$
4 **else if** OK **then**
5 $j \leftarrow 0$
6 **else** $j \leftarrow 0$
7 $k \leftarrow 0$
8 **repeat**
9 $j \leftarrow j + 1$
10 **until** $j > i$
11 **for** $k \leftarrow 0$ **to** i **do**
12 $j \leftarrow k$
13 **for** $k \leftarrow j$ **downto** 0 **step** 1 **do**
14 $i \leftarrow i$
15 **while** $k < i$ **do**
16 $j \leftarrow$ DOUBLE(k)
17 $k \leftarrow k + 1$
18 $j \leftarrow (j/2) + 1$
▷ Now $j = i$
19 **return** j
20 **End**

Figure 2: Example of commands, conditions and loops. On the left, the source code. On the right, the corresponding algorithm.

4 The configuration file

With this package you need the `ALgo.cfg` file. In this file you can specify your own keywords by adding a call to the `\LANG` call. You can also define the way you want your keywords and line numbers to be written by adding a call to the `\STYLE` command.

Important : The first argument of each of the two commands define the name of the style or the keywords set. It is this name you need to give as an option to use them.

The different possible values in the `\STYLE` command are:

- `BOLD`: keywords are written in bold font;
- `NUMBOLD`: line numbers are written in bold font;
- `ITALIC`: keywords are written in italic;
- `NUMITALIC`: line numbers are written in italic;
- `SMALLCAPS`: keywords are written in small caps (only possible without the `BOLD` and `ITALIC` options).

You can see examples in the `ALgo.cfg` file given in section 6.

Important : do not delete the `default` style and the `english` language. They are called by default.

Important : if you don't fill in all the possible fields in the `\LANG` command, the default english keywords will be used instead.

5 Examples

Figures 3-5 in this section present the same algorithm written with the ALgo package but with different options. Options used are given in the respective captions.

6 Annexe: example of ALgo.cfg file

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The default style                      %
% DO NOT DELETE IT!!!!                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\STYLE{default,
 BOLD,
 %ITALIC,
 %SMALLCAPS,
 NUMITALIC,
```

```
EgyptianMultiplication(i, j)
  ▷ Input: i integer greater than 0, j integer greater than 0
  ▷ Output: m equal to i × j
  ▷ Auxiliary: k integer
1 Begin
2   m ← 0
3   k ← 1
4   while 2 × k ≤ i do
5     k ← 2 × k
6     j ← 2 × j
7   while k ≥ 1 do
8     if k ≤ i then
9       m ← m + j
10      i ← i − k
11    k ← k div 2
12    j ← j div 2
13  return m
14 End
```

Figure 3: Algorithm written without any option. A label has been put line 4.

```
 %NUMBOLD
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The default language                           %
% DO NOT DELETE IT!!!!                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\LANG{english,
 BEGIN = Begin,
 END   = End,
 IF    = if,
 FI    = end~if,
 THEN  = then,
 ELSE  = else,
 FOR   = for,
 TO    = to,
 DOWNTO = downto,
 STEP  = step,
 DO    = do,
 DOWHILE = do,
 REPEAT = repeat,
 UNTIL = until,
```

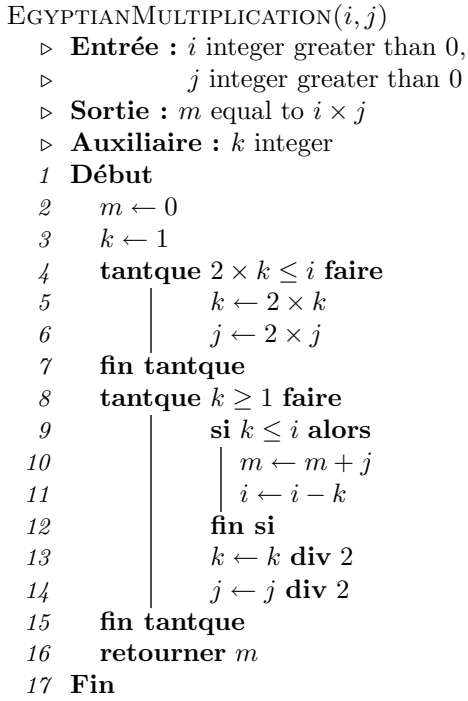

Figure 4: Algorithm written with the `french`, `rules`, `ends` and `noeqtab` options. A call to the \CUT command has been placed after the comma on the input line.

```
 ODFOR = end~for,
 WHILE = while,
 ODWHILE = end~while,
 RETURN = Return,
 BREAK = break,
 INPUT = Input:~,
 OUTPUT = Output:~,
 AUX = Auxiliary:~,
 TRUE = true,
 FALSE = false
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For algorithms in french                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\LANG{french,
 BEGIN = D\'ebut,
```

```
EgyptianMultiplication(i, j)
Input: i integer greater than 0, j integer greater than 0
Output: m equal to i × j
Auxiliary: k integer
Begin
  m ← 0
  k ← 1
  while 2 × k ≤ i do
  |  k ← 2 × k
  |  j ← 2 × j
  while k ≥ 1 do
  |  if k ≤ i then
  |  |  m ← m + j
  |  |  i ← i − k
  |  k ← k div 2
  |  j ← j div 2
  return m
End
```

Figure 5: Algorithm written with the `nonum` and `rules` options.

```
 END   = Fin,
 IF    = si,
 FI    = fin~si,
 THEN  = alors,
 ELSE  = sinon,
 FOR   = pour,
 TO    = \‘a,
 DOWNTO = \‘a,
 STEP  = pas,
 DO    = faire,
 DOWHILE = faire,
 REPEAT = r\’ep\’eter,
 UNTIL = jusqu’\‘a,
 ODFOR = fin~pour,
 WHILE = tantque,
 ODWHILE = fin~tantque,
 RETURN = Retourner,
 INPUT = Entr\’ee~:~,
 OUTPUT = Sortie~:~,
 AUX = Auxiliaire~:~,
 TRUE = vrai,
 FALSE = faux
}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For algorithms in german                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\LANG{german,
 BEGIN = Beginn,
 END   = Ende,
 IF    = wenn,
 FI    = Ende,
 THEN  = dann,
 ELSE  = sonst,
 FOR   = f\"ur,
 TO    = bis,
 DOWNTO = bis,
 STEP  = Schritt,
 DO    = tue,
 DOWHILE = tue,
 REPEAT = wiederhole,
 UNTIL = bis,
 ODFOR = Ende,
 WHILE = solange,
 ODWHILE = Ende,
 RETURN = Zur\"uck,
 INPUT = Eingabe~:~,
 OUTPUT = Ausgabe~:~,
 AUX = Daten~:~,
 TRUE = zutreffendes,
 FALSE = falsches
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For algorithms in spanish                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\LANG{spanish,
 BEGIN = Inicio,
 END   = Fin,
 IF    = si,
 FI    = fin~si,
 THEN  = entonces,
 ELSE  = en~caso~contrario,
 FOR   = desde,
 TO    = hasta,
 DOWNTO = hasta,
 STEP  = paso,
```

```
 DO    = hacer,
 DOWHILE = hacer,
 REPEAT = repetir,
 UNTIL = hasta~que,
 ODFOR = fin~desde,
 WHILE = mientras,
 ODWHILE = fin~mientras,
 RETURN = Vuelta,
 INPUT = Entrada~:~,
 OUTPUT = Salida~:~,
 AUX = Herramientas~:~,
 TRUE = verdadero,
 FALSE = falso
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For algorithms in portuguese           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\LANG{portuguese,
 BEGIN = In\'\i{}cio,
 END   = Fim,
 IF    = se,
 FI    = fim~se,
 THEN  = ent\~ao,
 ELSE  = sen\~ao,
 FOR   = para,
 TO    = at\'e,
 DOWNTO = at\'e,
 STEP  = etapa,
 DO    = fa\c{c}a,
 DOWHILE = fa\c{c}a,
 REPEAT = repita,
 UNTIL = at\'e,
 ODFOR = fim~para,
 WHILE = enquanto,
 ODWHILE = fim~enquanto,
 RETURN = Retorna,
 INPUT = Entrada~:~,
 OUTPUT = Sa\'\i{}da~:~,
 AUX = Dados~:~,
 TRUE = verdadeiro,
 FALSE = falso
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For algorithms in italian                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\LANG{italian,
 BEGIN = Inizio,
 END   = Fine,
 IF    = se,
 FI    = fine~se,
 THEN  = allora,
 ELSE  = altrimenti,
 FOR   = per,
 TO    = a,
 DOWNTO = a,
 STEP  = passo,
 DO    = fai,
 DOWHILE = fai,
 REPEAT = ripeti,
 UNTIL = fino~a,
 ODFOR = fine~per,
 WHILE = mentre,
 ODWHILE = fin~mentre,
 RETURN = Restituisci,
 INPUT = Input~:~,
 OUTPUT = Output~:~,
 AUX = Ausiliario~:~,
 TRUE = vero,
 FALSE = falso
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```