The algobox package
Typeset Algobox programs*

Julien "_FrnchFrgg_" Rivaud†

Released 2017/09/04

1 Documentation

1.1 Getting support

If you find bugs or keywords not recognized by the package, you can report it at the address https://gitlab.com/frnchfrgg-latex/algobox/issues.

1.2 Rationale

French mathematics and/or computer science teachers can use AlgoBox (http://www.xm1math.net/algobox/) to teach algorithmics to beginners. It uses tree-like indentation, fixed-form structures entered by buttons and dialogs rather than free text. This prevents most typos or small beginner mistakes that are often hard to detect.

AlgoBox can save a project in its own format, and export in a variety of formats, including LaTeX. Several problems arise:

- The output is not similar to what AlgoBox displays itself.
- The generated code is standalone and uses dependencies that might not be wanted, like algorithm and algpseudocode that can clash with other algorithm-like packages.
- The algorithm part of the generated code does not look like AlgoBox code and can thus be annoying to modify by hand if needed.

This small package intends to solve the problem by using a syntax very close to the real AlgoBox syntax and having a typeset output of the best quality possible, resembling the most to what AlgoBox displays in its editing window.

1.3 Simple usage

To convert an AlgoBox program to input suitable for this package you just have to:

1. Copy the program from the "Tester l'algorithme" window and remove the line numbers.

*This file describes v1.2a, last revised 2017/09/04.

†E-mail: frnchfrgg@free.fr

2. Add `\;` at the end of each line (or at the beginning of each line if like me you prefer to have those out of the way). The reason is that some constructs in AlgoBox programs depend on the line cuts whereas TEX treats those as normal spaces by default. Of course, one can setup TEX so that it temporarily obeys lines, but it *cannot work* if the algorithm is between braces (if you put in in a macro like `\fbox`).

3. Add a backslash at the beginning of all AlgoBox keywords, like `VARIABLES`, `SI`, `DEBUT_SINON` and so on.

4. Remove all underscores in AlgoBox keywords. You probably want to remove them in all cases because AlgoBox programs are not typeset in math mode and TEX will complain that you tried to use a subscript. If you really want an underscore, replace it by `\_` *but not in keywords* or they will not be recognized anymore.

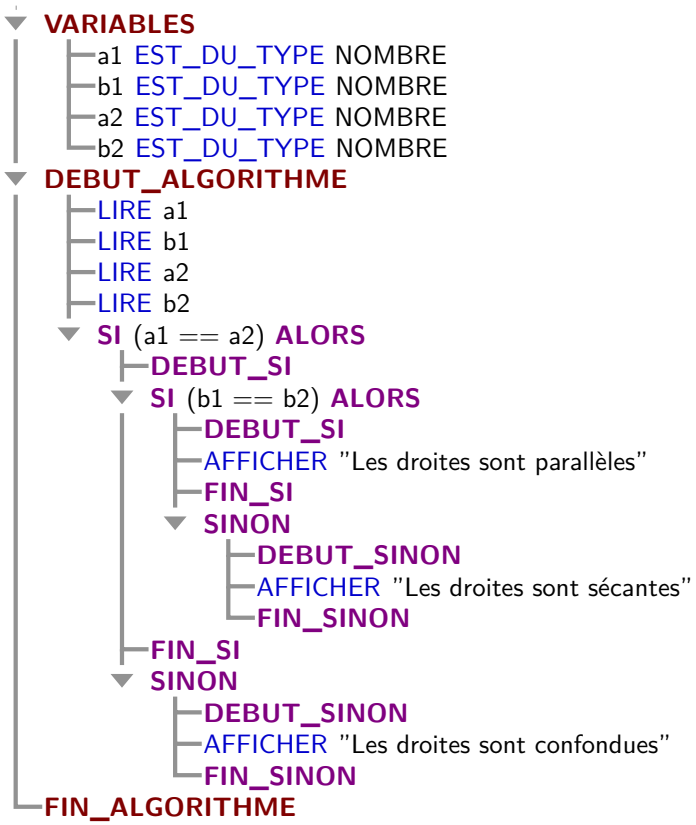
5. Put everything between `\begin{algobox}` and `\end{algobox}`.

Here is an example of the result:

```
\begin{algobox}
\;    \VARIABLES
\;      a1 \ESTDUTYPE NOMBRE
\;      b1 \ESTDUTYPE NOMBRE
\;      a2 \ESTDUTYPE NOMBRE
\;      b2 \ESTDUTYPE NOMBRE
\;    \DEBUTALGORITHME
\;      \LIRE a1
\;      \LIRE b1
\;      \LIRE a2
\;      \LIRE b2
\;    \SI (a1 == a2) \ALORS
\;      \DEBUTSI
\;      \SI (b1 == b2) \ALORS
\;        \DEBUTSI
\;        \AFFICHER "Les droites sont parallèles"
\;        \FINSI
\;        \SINON
\;          \DEBUTSINON
\;          \AFFICHER "Les droites sont sécantes"
\;          \FINSINON
\;      \FINSI
\;      \SINON
\;        \DEBUTSINON
\;        \AFFICHER "Les droites sont confondues"
\;        \FINSINON
\;  \FINALGORITHME
\end{algobox}
```

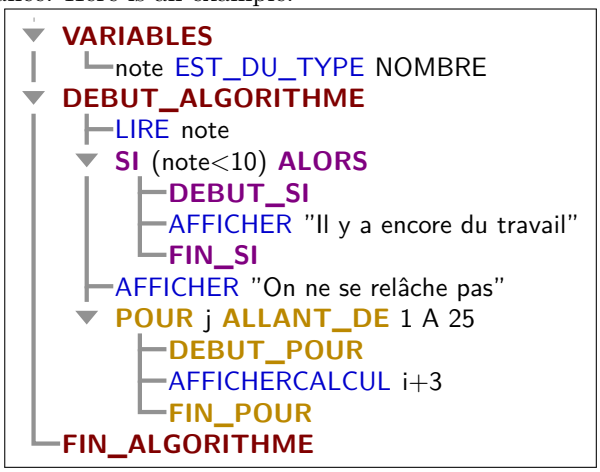
The spacing is only for your convenience as TEX treats any number of consecutive spaces as a single one.

Typesetting the previous example gives:



The algorithm is typeset as a TikZ picture, so is not breakable between pages.

You can of course put the environment in anything you deem fit, like a `\fbox` for instance. Here is an example:



1.4 Customizing the appearance

TODO

2 algobox implementation

```
⟨*package⟩
⟨@@=algobox⟩
\ProvidesExplPackage
  {\ExplFileName}{\ExplFileDate}{\ExplFileVersion}{\ExplFileDescription}
\RequirePackage{xparse}
\RequirePackage{environ}
\RequirePackage{tikz}

\ExplSyntaxOff
\usetikzlibrary{calc}
\ExplSyntaxOn

\RequirePackage{xcolor}

\int_new:N \l__algobox_level_int

```

The colors are defined in `algohighlighter.cpp` in the AlgoBox source, except for the tree lines color. The names here are the same, with `algobox` prepended.

```
\definecolor{algoboxarbre}{RGB}{146,146,146}
\definecolor{algoboxColorStandard}{HTML}{000000}
\definecolor{algoboxColorComment}{HTML}{606060}
\definecolor{algoboxColorBloc}{HTML}{800000}
\definecolor{algoboxColorCommande}{HTML}{0000CC}
\definecolor{algoboxColorSi}{HTML}{800080}
\definecolor{algoboxColorTantQue}{HTML}{DD6F06}
\definecolor{algoboxColorPour}{HTML}{BB8800}
\definecolor{algoboxColorFonction}{HTML}{9A4D00}

\tikzset{
    every~algobox/.style=,
    every~smalgobox/.style={baseline=(block.base)},
    algobox~ligne/.style={
        xshift=1em, text~depth=0pt,
        inner~xsep=0.1ex, inner~ysep=0.5ex
    },
    algobox~arbre/.style={ultra~thick, color=algoboxarbre},
    algobox~indentsymb/.style={x=1ex, y=0.6ex, color=algoboxarbre},
    algobox~indent/.style={xshift=2em},
    algobox~normal/.style={
        font=\sffamily,
        color=algoboxColorStandard,
    },
    algobox~commentaire/.style={
        algobox~normal/.try,
        color=algoboxColorComment,
    },
    algobox~commande/.style={
        algobox~normal/.try,
        color=algoboxColorCommande,
    },
```

```
    algobox~structure/.style={
        font=\sffamily\bfseries,
        color=algoboxColorBloc
    },
    algobox~sialors/.style={
        algobox~structure/.try,
        color=algoboxColorSi,
    },
    algobox~pour/.style={
        algobox~structure/.try,
        color=algoboxColorPour,
    },
    algobox~tantque/.style={
        algobox~structure/.try,
        color=algoboxColorTantQue,
    },
    algobox~fonctions/.style={
        algobox~structure/.try,
        color=algoboxColorFonction,
    },
}

```

2.1 Formatting commands

```


\tl_new:N \l__algobox_lastnode_tl
\cs_new_protected:Nn \__algobox_block:nn {
    \node[algobox~ligne/.try,
        \tl_if_empty:nF{#1}{algobox~#1/.try},
        anchor=north~west
    ]
        (block) at (A\int_use:N \l__algobox_level_int |- H)
        {\vphantom{A}#2};
    \draw[algobox~arbre/.try]
        (A\int_use:N \l__algobox_level_int)
        -- (A\int_use:N \l__algobox_level_int |- block.west)
                coordinate (A\int_use:N \l__algobox_level_int)
        -- (block.west);
    \coordinate (H) at (block.south);
    \tl_set:Nn \l__algobox_lastnode_tl {block}
}

\cs_new_protected:Nn \__algobox_indent:nn {
    \node[algobox~ligne/.try,
        \tl_if_empty:nF{#1}{algobox~#1/.try},
        anchor=north~west
    ]
        (block) at (A\int_use:N \l__algobox_level_int |- H)
        {\vphantom{A}#2};
    \draw[algobox~arbre/.try]
        (A\int_use:N \l__algobox_level_int)
        -- (A\int_use:N \l__algobox_level_int |- H);
    \fill[algobox~indentsymb/.try]
```

```
        (A\int_use:N \l__algobox_level_int |- block.west)
        +(0,-1) -- +(1,1) -- +(-1,1) -- cycle;
    \coordinate (H) at (block.south);
    \coordinate (T) at (A\int_use:N \l__algobox_level_int |- H);
    \coordinate (A\int_use:N \l__algobox_level_int) at (T);
    \int_incr:N \l__algobox_level_int
    \coordinate[algobox~indent/.try] (A\int_use:N \l__algobox_level_int) at (T);
    \tl_set:Nn \l__algobox_lastnode_tl {block}
}
\cs_new_protected:Nn \__algobox_unindent: {
    \int_decr:N \l__algobox_level_int
}

\cs_new_protected:Nn \__algobox_node:nn {
    \node[algobox~#1/.try, anchor=base~west, inner~xsep=0pt]
        (node) at (\l__algobox_lastnode_tl.base~east) {#2};
    \tl_set:Nn \l__algobox_lastnode_tl {node}
}

```

2.2 Parsing helpers

```


\cs_new_protected:Nn \__algobox_parseall:n {
    \tl_set:Nn \l_tmpa_tl {#1}
    \seq_set_split:NnV \l_tmpa_seq {\;} \l_tmpa_tl
    % Now build up the algorithm body, working around the few infix notations.
    \tl_clear:N \l__algobox_body_tl
    \seq_map_variable:NNn \l_tmpa_seq \l_tmpa_tl {
        \tl_if_in:NnT \l_tmpa_tl {\ESTDUTYPE} {
            \tl_put_left:Nn \l_tmpa_tl {\__algobox_ESTDUTYPE:w}
        }
        \tl_if_in:NnT \l_tmpa_tl {\PRENDLAVALEUR} {
            \tl_put_left:Nn \l_tmpa_tl {\__algobox_PRENDLAVALEUR:w}
        }
        \tl_put_right:Nn \l_tmpa_tl { \; }
        \tl_put_right:NV \l__algobox_body_tl \l_tmpa_tl
    }
    \tl_use:N \l__algobox_body_tl
}

\bool_new:N \l__algobox_balanced_first_bool
\cs_new_protected:Npn \__algobox_new_balanced_command:nNNnn #1#2#3 {
    \cs_new_protected:cpn {#1} ##1 #2 {
        \tl_set:Nn \l__algobox_prefix_tl { ##1 }
        \tl_clear:N \l__algobox_balanced_tl
        \bool_set_true:N \l__algobox_balanced_first_bool
        \use:c {#1_stage2} #2
    }
    \cs_new_protected:cpn {#1_stage2} ##1 #2 ##2 #3 {
        \tl_put_right:Nn \l__algobox_balanced_tl { ##1 }
        \bool_if:NF \l__algobox_balanced_first_bool {
            \tl_put_right:Nn \l__algobox_balanced_tl { #2 }
        }
```

```
        \tl_if_in:nnTF {##2} {#2} {
            \bool_set_false:N \l__algobox_balanced_first_bool
            \use:c {#1_stage2} ##2 \q_mark
        }{
            \tl_put_right:Nn \l__algobox_balanced_tl { ##2 }
            \tl_replace_all:Nnn \l__algobox_balanced_tl {\q_mark} {#3}
            \tl_set:Nx \l_tmpa_tl {
                \exp_not:c {#1_stage3}
                \exp_not:V \l__algobox_prefix_tl
                \exp_not:n {#2}
                {
                    \exp_not:V \l__algobox_balanced_tl
                }
            }
            \tl_use:N \l_tmpa_tl
        }
    }
    \exp_args:Nc \NewDocumentCommand {#1_stage3}
}

\cs_new_protected:Npn \__algobox_if_next:NTF #1#2#3 {
    \peek_meaning_remove_ignore_spaces:NTF \; {
        \__algobox_if_next:NTF #1 {#2} {#3}
    }{
        \peek_meaning_remove_ignore_spaces:NTF \par {
            \__algobox_if_next:NTF #1 {#2} {#3}
        }{
            \peek_meaning_ignore_spaces:NTF #1 {#2} {#3}
        }
    }
}
```

2.3 Parsing/typesetting commands

```

\NewDocumentCommand \__algobox_ESTDUTYPE:w { +u{\ESTDUTYPE} +u{\;} } {
    \__algobox_block:nn {normal} {
        \ignorespaces #1 \unskip\c_space_token\null
    }
    \__algobox_node:nn {commande}{EST_DU_TYPE}
    \__algobox_node:nn {normal}{\null\c_space_token#2}
}

\NewDocumentCommand \__algobox_FINALGORITHME:w {} {
    \int_while_do:nNnn \l__algobox_level_int > \c_zero {
        \__algobox_unindent:
    }
    \__algobox_block:nn {structure} {FIN_ALGORITHME}
}

\NewDocumentCommand \__algobox_PRENDLAVALEUR:w { u{\PRENDLAVALEUR} m } {
    \__algobox_block:nn {normal} {
        \ignorespaces#1\unskip\c_space_token\null
    }
    \__algobox_node:nn {commande}{PREND_LA_VALEUR}
```

```
    \__algobox_node:nn {normal}{\null\c_space_token#2}
}

\__algobox_new_balanced_command:nNNnn
            {__algobox_SI:w} \DEBUTSI \FINSI { r() u{\DEBUTSI} +m }
{
    \__algobox_indent:nn {sialors} {SI}
    \__algobox_node:nn {normal}{\null\c_space_token(#1)\c_space_token\null}
    \__algobox_node:nn {sialors}{ALORS}
    \__algobox_block:nn {sialors} {DEBUT_SI}
    #3
    \__algobox_block:nn {sialors} {FIN_SI}
    \__algobox_if_next:NTF \SINON {
        \__algobox_SINON:w
    }{
        \__algobox_unindent:
    }
}
\__algobox_new_balanced_command:nNNnn
            {__algobox_SINON:w} \DEBUTSINON \FINSINON { u{\DEBUTSINON} +m } {
    \__algobox_indent:nn {sialors} {SINON}
    \__algobox_block:nn {sialors} {DEBUT_SINON}
    #2
    \__algobox_block:nn {sialors} {FIN_SINON}
    \__algobox_unindent:
    \__algobox_unindent:
}

\__algobox_new_balanced_command:nNNnn
            {__algobox_POUR:w} \DEBUTPOUR \FINPOUR
            { u{\ALLANTDE} u{\A} u{\DEBUTPOUR} +m } {
    \__algobox_indent:nn {pour} {POUR}
    \__algobox_node:nn {normal}{\null\c_space_token #1\unskip\c_space_token\null}
    \__algobox_node:nn {pour}{ALLANT_DE}
    \__algobox_node:nn {normal}{\null\c_space_token #2\unskip
                        \c_space_token A~#3\unskip\c_space_token\null}
    \__algobox_block:nn {pour} {DEBUT_POUR}
    #4
    \__algobox_block:nn {pour} {FIN_POUR}
    \__algobox_unindent:
}

\__algobox_new_balanced_command:nNNnn {__algobox_TANTQUE:w} \DEBUTTANTQUE \FINTANTQUE
        { r() u{\DEBUTTANTQUE} +m }
{
    \__algobox_indent:nn {tantque} {TANT_QUE}
    \__algobox_node:nn {normal}{\null\c_space_token(#1)\c_space_token\null}
    \__algobox_node:nn {tantque}{FAIRE}
    \__algobox_block:nn {tantque} {DEBUT_TANT_QUE}
    #3
    \__algobox_block:nn {tantque} {FIN_TANT_QUE}
    \__algobox_unindent:
}

```

```
\NewDocumentCommand \__algobox_FONCTION:w { +u{\;} } {
    \int_while_do:nNnn \l__algobox_level_int > \c_one {
        \__algobox_unindent:
    }
    \__algobox_indent:nn {fonctions} {FONCTION}
    \__algobox_node:nn {normal}{\null\c_space_token#1}
}

\NewDocumentCommand \__algobox_LINE:w { s O{normal} m } {
    \IfBooleanTF {#1} {
        \__algobox_indent:nn {#2} {#3}
        \__algobox_unindent:
    }{
        \__algobox_block:nn {#2} {#3}
    }
}

\NewDocumentCommand \__algobox_NODE:w { O{normal} m } {
    \__algobox_node:nn {#1} {#2}
}
```

2.4 User commands and environment

```

\cs_new_protected_nopar:Nn \__algobox_make_keyword:n {
    \tl_set:Nn \l_tmpa_tl {#1}
    \tl_replace_all:Nnn \l_tmpa_tl {_} {}
    \cs_set:cpn {\l_tmpa_tl} {\__algobox_keyword:nw {#1}}
}

\NewDocumentCommand \__algobox_keyword:nw {m +u{\;}} {
    \__algobox_block:nn {commande} {#1}
    \__algobox_node:nn {normal}{\null\c_space_token#2}
}

\cs_new_protected_nopar:Nn \__algobox_make_structure:nn {
    \tl_set:Nn \l_tmpa_tl {#2}
    \tl_replace_all:Nnn \l_tmpa_tl {_} {}
    \cs_set:cpn {\l_tmpa_tl} {\__algobox_topstructure:nn {#1}{#2}}
}

\cs_new_protected_nopar:Nn \__algobox_topstructure:nn {
    \int_while_do:nNnn \l__algobox_level_int > \c_zero {
        \__algobox_unindent:
    }
    \__algobox_indent:nn {#1} {#2}
}

\cs_new_protected_nopar:Nn \__algobox_make_func_structure:nn {
    \tl_set:Nn \l_tmpa_tl {#1}
    \tl_replace_all:Nnn \l_tmpa_tl {_} {}
    \cs_set:cpn {\l_tmpa_tl} {\__algobox_funcstructure:nn {#1}{#2}}
}

```

```
\cs_new_protected_nopar:Nn \__algobox_funcstructure:nn {
    \int_while_do:nNnn \l__algobox_level_int > \c_two {
        \__algobox_unindent:
    }
    \bool_if:nTF {#2}{\__algobox_indent:nn}{\__algobox_block:nn} {fonctions} {#1}
}

\clist_const:Nn \c__algobox_keywords_clist {
    PAUSE, LIRE, AFFICHERCALCUL, AFFICHER,
    TRACER_POINT, TRACER_POINT_Rouge, TRACER_POINT_Vert,
    TRACER_POINT_Bleu, TRACER_POINT_Blanc,
    TRACER_SEGMENT, TRACER_SEGMENT_Rouge, TRACER_SEGMENT_Vert,
    TRACER_SEGMENT_Bleu, TRACER_SEGMENT_Blanc,
    RENVOYER, APPELER_FONCTION
}

\cs_new_protected_nopar:Nn \__algobox_begincode:n {
    \int_zero:N \l__algobox_level_int
    \begin{tikzpicture}[#1]
        \coordinate (H) at (0,0);
        \coordinate (A\int_use:N \l__algobox_level_int) at (0,0);
        \cs_set_eq:NN \par \prg_do_nothing:
        \cs_set_eq:NN \; \prg_do_nothing:
        \cs_set_eq:NN \LINE             \__algobox_LINE:w
        \cs_set_eq:NN \NODE             \__algobox_NODE:w
        \cs_set_eq:NN \FINALGORITHME    \__algobox_FINALGORITHME:w
        \cs_set_eq:NN \SI               \__algobox_SI:w
        \cs_set_eq:NN \POUR             \__algobox_POUR:w
        \cs_set_eq:NN \TANTQUE          \__algobox_TANTQUE:w
        \cs_set_eq:NN \FONCTION         \__algobox_FONCTION:w
        \cs_set:Npn \//##1\;            {\__algobox_block:nn{commentaire}{//##1}}
        \clist_map_function:NN \c__algobox_keywords_clist \__algobox_make_keyword:n
        \__algobox_make_structure:nn {structure} {VARIABLES}
        \__algobox_make_structure:nn {structure} {DEBUT_ALGORITHME}
        \__algobox_make_structure:nn {fonctions} {FONCTIONS_UTILISEES}
        \__algobox_make_func_structure:nn {VARIABLES_FONCTION} \c_true_bool
        \__algobox_make_func_structure:nn {DEBUT_FONCTION} \c_false_bool
        \__algobox_make_func_structure:nn {FIN_FONCTION} \c_false_bool
}
\cs_new_protected_nopar:Nn \__algobox_closecode: {
    \end{tikzpicture}
}



\NewDocumentCommand \smalgobox { O{} m } {
    \group_begin:
    \__algobox_begincode:n {every~smalgobox/.try,#1}
    \__algobox_parseall:n {#2}
    \__algobox_closecode:
    \group_end:
}

\NewDocumentEnvironment {algobox} {O{}} {
```

```
    \cs_set_protected:Nn \__algobox_algobox:n {
        \__algobox_begincode:n {every~algobox/.try,#1}
        \__algobox_parseall:n {##1}
        \__algobox_closecode:
    }
    \Collect@Body\__algobox_algobox:n
}{
}
```
⟨/package⟩