

Package
listofitems

v1.6
1 November 2018

Christian TELLECHEA*
Steven B. SEGLETES†

This simple package is designed to read a list of items whose parsing separator may be selected by the user. Once the list is read, its items are stored in a structure that behaves as a dimensioned array. As such, it becomes very easy to access an item in the list by its number. For example, if the list is stored in the macro `\foo`, the item number 3 is designated by `\foo[3]`.

A component may, in turn, be a list with a parsing delimiter different from the parent list, paving the way for nesting and employing a syntax reminiscent of an array of several dimensions of the type `\foo[3,2]` to access the item number 2 of the list contained within the item number 3 of the top-tier list.

*unbonpetit@netc.fr

†steven.b.segletes.civ@mail.mil

1 Preface

This package loads no external packages, must be used with the ε -TeX engine, and must be called in (pdf)(Xe)(lua)TeX with the invocation

```
\usepackage{listofitems}
```

and under (pdf)(Xe)(Lua)TeX by way of

```
\input listofitems.tex
```

2 Read a Simple List

Set the parsing separator The default parsing separator is the comma and if we want change it, we must do so before reading a list of items, with the definition `\setsepchar{< parsing-separator >}`. A *< parsing-separator >* is a set of tokens which possess catcodes different from 1 and 2 (the opening and closing braces), 14 (usually %) and 15. The token of catcode 6 (usually #) is accepted only if it is followed by an integer, denoting the argument of a macro; In no case should this token be provided alone as the *< parsing-separator >*. Commands can be included in this set of tokens, including the TeX primitive `\par`.

The parsing-separator *< delimiter >* “/” is reserved by default for nested lists (see page 3). It is therefore not proper to write “`\setsepchar{/}`” because the listofitems package would misunderstand that you want to read a nested list. To set “/” as the *< parsing-separator >* for a simple list, it is necessary, using the optional argument, to choose a different parsing-separator *< delimiter >* for nested lists, for example “.”, and write “`\setsepchar[.]{/}`”.

It is not possible to select | as the *< delimiter >* because it would conflict with the logical **OR**, denoted “||” (see below). However, one can work around this limitation, at one’s own peril, writing “`\setsepchar{||}`”.

Read a list To read the list of items, the `\readlist<macro-list>{<list>}` should be called. In so doing, the *<list>* is read and the items are stored in a macro, denoted *<macro-list>* which therefore acts as a table with the items of the *<list>*. If braces appear as part of a list item, they *must* be balanced. Tokens possessing the catcodes 6, 14 and 15 are not allowed in the lists.

For example, to set the *<macro-list>* named `\foo`, we can write

```
\setsepchar{,}
\readlist\foo{12,abc,x y ,{\bfseries z},,\TeX,,!}
```

If the *<list>* is contained in a macro, then this macro is expanded. Therefore, we can simply employ the syntax `\readlist<macro-list>{<macro>}` as in

```
\setsepchar{,}
\def\List{12,abc,x y ,{\bfseries z},,\TeX,,!}
\readlist\foo\List
```

The macro `\greadlist` makes *global* assignments and therefore, enables the use of *<macro-list>* outside of the group where `\greadlist` has been executed.

Access an item The macro `\foo` requires a numeric argument in square brackets, which we symbolically denote as *i*, indicating the rank of the item you wish to access. So `\foo[1]` is³ “12”. Similarly, `\foo[4]` is “`{\bfseries z}`”.

The number *i* can also be negative in which case the counting is done from the end of the list: `-1` represents the last item, `-2` the penultimate, etc. If the number of items is *n*, then the argument `-n` is the first item.

³`\foo[i]` requires 2 expansions to give the item.

In general, if a $\langle list \rangle$ has a length n , then the index i can be in the interval $[1; n]$ or $[-n; -1]$. Otherwise, a compilation error occurs.

If the index is empty, $\foo[]$ produces the complete $\langle list \rangle$.

The macro \foosep is created. It is used with the syntax $\foosep[\langle index \rangle]$ and allows access to the parsing-separator that follows the item of rank $\langle index \rangle$. The last parsing-separator (the one following the last item) is empty. If the $\langle index \rangle$ is empty, $\foosep[]$ is empty.

Select several possible parsing separators To specify several possible separators, use the **OR** operator, denoted “|”. One can use this feature, for example, to isolate the terms in an algebraic sum:

```
\setsepchar{+|-}
\readlist\term{17-8+4-11}
1) \term[1] (parsing separator = \termsep[1])\par
2) \term[2] (parsing separator = \termsep[2])\par
3) \term[3] (parsing separator = \termsep[3])\par
4) \term[4] (parsing separator = \termsep[4])
```

Number of items If we write $\readlist\langle macro-list \rangle\{\langle list \rangle\}$, then the macro $\langle macro-list \rangle len$ contains⁴ the number of the items in $\langle list \rangle$. In the example with \foo , the macro \foolen expands to 8.

View all items For purposes of debugging, the macro $\showitems\langle macro-list \rangle$ includes all items from a list, while the star version displays these items “detokenized.”⁵

```
\showitems\foo\par
\showitems*\foo
```

The presentation of each list item is assigned to the macro \showitemsmacro whose code is

```
\newcommand\showitemsmacro[1]{%
  \begingroup\fbboxsep=0.25pt \fbboxrule=0.5pt \fbbox{\strut#1}\endgroup
  \hskip0.25em\relax}
```

It is therefore possible – and desirable – to redefine it if we desire a different presentation effect.

The macro \fbbox and associated dimensions \fbboxsep and \fbboxrule , are defined by $listofitems$, when *not* compiled under \LaTeX , to achieve the same result *as if* performed under \LaTeX .

Suppression of extreme (leading/trailing) spaces By default, $listofitems$ reads and retains the spaces located at the beginning and end of an item. For these spaces to be ignored when reading the $\langle list \rangle$, execute the starred version $\readlist*\langle macro \rangle\{\langle list \rangle\}$:

```
\setsepchar{,}
\readlist*\foo{12,abc, x y ,{\bfseries z}, ,\TeX,,!}
\showitems\foo
```

Managing empty items By default, the $listofitems$ package retains and accounts for empty items. Thus, in the previous example, the 2nd expansion of $\foo[7]$ is empty. For empty items of the list (i.e., those list items defined by two consecutive parsing delimiters) to be ignored, we must, before invoking \readlist , execute the macro \ignoreemptyitems . To return to the default package behavior, simply execute the macro \reademptyitems .

⁴That is to say, it is purely expandable and grows into a number.
⁵The primitive \detokenize , conducting this decomposition, inserts a space after each control sequence.

This option can be used alone or in combination with `\readlist*`, in which case the suppression of extreme (leading/trailing) spaces occurs *before* listofitems ignores the empty list items:

<pre>\setsepchar{,} \ignoreemptyitems \readlist\foo{12,abc, x y ,{\bfseries z}, ,\TeX,,!} a) number of items = \foolen\par \showitems\foo \readlist*\foo{12,abc, x y ,{\bfseries z}, ,\TeX,,!} b) number of items = \foolen\par \showitems\foo</pre>	<pre>a) number of items = 7 12 abc x y z TeX ! b) number of items = 6 12 abc x y z TeX !</pre>
---	--

Iterate over a list Once a list read by `\readlist` and stored in a $\langle macro-list \rangle$, one may iterate over the list with the syntax `\foreachitem $\langle variable \rangle$ \in $\langle macro-list \rangle$ { $\langle code \rangle$ }`: The $\langle variable \rangle$ is a macro chosen by the user that will loop over the value of each item in the list. The macro $\langle variable \rangle$ cnt represents the sequence number of the item in $\langle variable \rangle$.

<pre>\setsepchar{ }% parsing-separator = space \readlist\phrase{One phrase to test.} \foreachitem\word\in\phrase{List item number \wordcnt{}: \word\par}</pre>	<pre>List item number 1: One List item number 2: phrase List item number 3: to List item number 4: test.</pre>
--	--

Assign an item to a macro The `\itemtomacro $\langle macro-list \rangle$ [$\langle index \rangle$] $\langle macro \rangle$` assigns to the $\langle macro \rangle$ the item designated by $\langle macro-list \rangle$ [$\langle index \rangle$]. The $\langle macro \rangle$ thus defined is purely expandable provided that the tokens in the items are expandable.

<pre>\setsepchar{ }% parsing-separator = space \readlist\phrase{One phrase to test.} \itemtomacro\phrase[2]\aword \meaning\aword\par \itemtomacro\phrase[-1]\wordattheend \meaning\wordattheend</pre>	<pre>macro:->phrase macro:->test.</pre>
---	---

3 Nested Lists

We speak of a list being “nested” when asking listofitems to read a list where the items are, in turn, understood as being a list (implying a parsing separator different from the top-tier list). The nesting depth is not limited, but in practice, a depth of 2 or 3 will usually suffice.

Defining the parsing separators To indicate that a list will be nested, so that the list parsing will be performed recursively, one must specify multiple parsing separators, each corresponding to the particular tier of nesting. This list of parsing separators is itself given as a delimited list to the macro `\setsepchar`, with the syntax `\setsepchar[$\langle delimiter \rangle$]{ $\langle delimited-list-of-parsing-separators \rangle$ }`. By default, the $\langle delimiter \rangle$ is “/”. Thus, writing

```
\setsepchar{\\,/ }
```

indicates a recursive depth of 3, with the parsing-separator list delimiter defaulting to “/”:

- Tier 1 items are parsed between “\” delimiters;
- Tier 2 items are found within Tier 1 items, parsed between “,” delimiters;
- finally, the Tier 3 items are found within Tier 2 items, parsed between the “_” delimiters.

The $\langle depth \rangle$ of nesting is contained in the purely expandable macro `\nestdepth`.

Read and access list items For nested lists, the use of indices obey the following rules:

- [] is the main list, i.e., the argument of `\readlist`;
- [*i*] means the item number *i* of the main list;
- [*i*, *j*] means the item number *j* of the list mentioned in the previous point (a subitem);
- [*i*, *j*, *k*] means the item number *k* of the list mentioned in the previous point (a sub-subitem);
- etc.

As in the case of a non-nested list, the index may be negative.

To read items, the syntax of `\readlist` is exactly the same as that for simple (non-nested) lists:

<pre>\setsepchar{\\,/ } \readlist\baz{1,2 a b,3 c\\4 d e f,5,6\\7,,8, ,9 xy z} a) \string\baz[1] is \baz[1]\par b) \string\baz[1,1] is \baz[1,1]\par c) \string\baz[1,1,1] is \baz[1,1,1]\par b) \string\bar[1,2] is \baz[1,2]\par e) \string\baz[1,2,3] is \baz[1,2,3]\par f) \string\baz[-2,1,-1] is \baz[-2,1,-1]</pre>	<pre>a) \baz[1] is 1,2 a b,3 c b) \baz[1,1] is 1 c) \baz[1,1,1] is 1 b) \bar[1,2] is 2 a b e) \baz[1,2,3] is b f) \baz[-2,1,-1] is f</pre>
--	--

The operator “||” This operator may be employed at any level of nesting.

<pre>\setsepchar[,]{+ -,* /} \readlist\numbers{1+2*3-4/5*6} Term 1: \numbers[1]\par Term 2: \numbers[2] (factors: \numbers[2,1] and \numbers[2,2])\par Term 3: \numbers[3] (factors: \numbers[3,1], \numbers[3,2] and \numbers[3,3])</pre>	<pre>Term 1: 1 Term 2: 2*3 (factors: 2 and 3) Term 3: 4/5*6 (factors: 4, 5 and 6)</pre>
--	---

Number of list items The macro `\listlen<macro-list>[<index>]` requires 2 expansions in order to give the number of items in the list specified by the *<index>*. The *<depth>* of the *<index>* must be strictly less than that of the list.

For the case where the *<index>* is empty, `\listlen<macro-list>[]`, with 2 expansions, yields the identical result as `<macro-list>len` with 1 expansion.

<pre>\setsepchar{\\,/ } \readlist\baz{1,2 a b,3 c\\4 d e f,5,6\\7,,8, ,9 xy z} a) \bazlen\ or \listlen\baz[]\par b) \listlen\baz[1]\par c) \listlen\baz[2]\par d) \listlen\baz[3]\par e) \listlen\baz[3,1]\par f) \listlen\baz[3,4]\par% 2 empty items g) \listlen\baz[3,5]</pre>	<pre>a) 3 or 3 b) 3 c) 3 d) 5 e) 1 f) 2 g) 3</pre>
---	--

Displaying list items The macro `\showitems<macro-list>[<index>]` displays items from the list specified by *<index>*, in the same manner as `\listlen`. The *<depth>* of the *<index>* must be strictly less than that of the *<list>*.

<pre>\setsepchar{\\,/ } \readlist\baz{1,2 a b,3 c\\4 d e f,5,6\\7,,8, ,9 xy z} a) \showitems\baz[]\par b) \showitems\baz[1]\par c) \showitems\baz[2]\par d) \showitems\baz[3]\par e) \showitems\baz[3,1]\par f) \showitems\baz[3,4]\par% 2 empty items g) \showitems\baz[3,5]</pre>	<pre>a) 1,2 a b,3 c 4 d e f,5,6 7,,8, ,9 xy z b) 1 2 a b 3 c c) 4 d e f 5 6 d) 7 8 9 xy z e) 7 f) g) 9 xy z</pre>
---	---

Empty items and extreme (leading/trailing) spaces The removal of empty items and/or leading/trailing spaces will occur in *all* the items, regardless of the degree of nesting. It is clear that a space, “_”, is useless as a parsing separator if you want to use `\readlist*`. Therefore, in the following example, “*” is instead selected as the (3rd-tier) parsing separator.

Further, we remove only the extreme spaces, but retain empty items.

<pre> \setsepchar{\,/,*} \readlist*\baz{1, 2*a*b ,3*c\4*d*e*f,5,6\7,,8, ,9* xy *z} a) \showitems\baz[\par b) \showitems\baz[1]\par c) \showitems\baz[2]\par d) \showitems\baz[3]\par e) \showitems\baz[3,1]\par f) \showitems\baz[3,4]\par g) \showitems\baz[3,5]% "xy" without extreme spaces </pre>	<pre> a) 1, 2*a*b ,3*c 4*d*e*f,5,6 7,,8, ,9* xy *z b) 1 2*a*b 3*c c) 4*d*e*f 5 6 d) 7 8 9* xy *z e) 7 f) g) 9 xy z </pre>
---	---

Iterate over a list The syntax `\foreachitem <variable> \in <macro>[<index>]{<code>}` remains valid where now the `<index>` specifies the item (understood as a list) on which to iterate. The `<depth>` of the `<index>` must be strictly less than that of the `<list>`.

Assign an item to a macro The syntax `\itemtomacro<macro-list>[<index>]<macro>` remains valid to assign to `<macro>` the item specified by `<macro-list>[<index>]`.

<pre> \setsepchar[,]{\, } \readlist\poem{There once was a runner named Dwight\% Who could speed even faster than light.\% He set out one day\% In a relative way\% And returned on the previous night.} \itemtomacro\poem[2]\verse 2nd verse = \verse \itemtomacro\poem[2,-4]\word A word = \word </pre>	<pre> 2nd verse = Who could speed even faster than light. A word = even </pre>
---	--

The macro `\itemtomacro` makes a global assignment.

4 Balanced Tokens

For the parsing of items, it is possible, with version 1.6, to take into account the presence of *balanced tokens*. Thus, if a list of paired tokens is defined, then each parsed item in the list will extend to the first `<separator>`, while assuring that any paired tokens are balanced (i.e., occur in matched pairs within the item).

To define a list of balanced-token pairs, we use

```
\defpair{<tok1><tok2><tok3><tok4>...}
```

where the token list is read in pairs to form each matched-token pair. A `<token>` that serves within a matched pair must consist of a single character—macros, primitives, spaces, braces, the token “#”, as well as sets of several-tokens-between-braces are all forbidden. The two tokens which form a pair *must* be different from each other.

<pre> \setsepchar{+ -} \defpair{()} \readlist\terms{1+2*[3+4*(5+6-7)+8]-9+10} \showitems\terms </pre>	<pre> 1 2*[3+4*(5+6-7)+8] 9 10 </pre>
--	---------------------------------------

To return to the package's default behavior, that is, without paired tokens, you must execute

```
\defpair{}
```

In an expression, in order to store in a macro that which is between two matched tokens, we can call on

```
\insidepair<tok1><tok2>{<expression>}\macro
```

which will put in the `\macro` that which lies between the pair `<tok1>``<tok2>` in the `<expression>`.

```
\setsepchar{+|-}
\defpair{()}
\readlist\terms{1+2*(3+4*(5+6-7)+8)-9+10}
\showitems\terms

\itemtomacro\terms[2]\parenterm
In the outer parenthesis:
\insidepair()\parenterm\inbigparen
"\inbigparen"

In the inner parenthesis:
\insidepair()\inbigparen\insmallparen
"\insmallparen"
```

$$1 \ 2^*(3+4*(5+6-7)+8) \ 9 \ 10$$

In the outer parenthesis: "3+4*(5+6-7)+8"
 In the inner parenthesis: "5+6-7"

5 The Code

Any suggestion, bug report, remark, request, addition or modification of functionality is welcome; in this case, I invite users of listofitems to send me an email to unbonpetit@netc.fr.

The code below is the exact verbatim of the file `listofitems.tex`. I hope that the few comments scattered throughout it will be enough for the user or the curious to understand the internal machinery of this package:

```
1 % !TeX encoding = ISO-8859-1
2 % Ce fichier contient le code de l'extension "listofitems"
3 %
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 %
6 \def\loiname           {listofitems}
7 \def\loiver            {1.6}
8 %
9 \def\loidate           {2018/11/01}
10 %
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12 %
13 % Author       : Christian Tellechea, Steven B. Segletes
14 % Status      : Maintained
15 % Maintainer  : Christian Tellechea
16 % Email       : unbonpetit@netc.fr
17 %              steven.b.segletes.civ@mail.mil
18 % Package URL: https://www.ctan.org/pkg/listofitems
19 % Bug tracker: https://framagit.org/unbonpetit/listofitems/issues
20 % Repository  : https://framagit.org/unbonpetit/listofitems/tree/master
21 % Copyright   : Christian Tellechea 2016-2018
22 % Licence     : Released under the LaTeX Project Public License v1.3c
23 %              or later, see http://www.latex-project.org/lppl.txt
24 % Files      : 1) listofitems.tex
25 %              2) listofitems.sty
26 %              3) listofitems-fr.tex
27 %              4) listofitems-fr.pdf
```

```

28 %          5) listofitems-en.tex          %
29 %          6) listofitems-en.pdf        %
30 %          7) README                     %
31 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32 \ifdefined\ProvidesPackage\else
33 \immediate\write -1 {%
34 Package: \loidate\space v\loiver\space Grab items in lists using user-specified sep char (CT)}%
35 \fi
36
37 \expandafter\edef\csname loi_restorecatcode\endcsname{\catcode\number'\_=\number\catcode'\_ \relax}
38 \catcode'\_11
39
40 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% gestion des erreurs %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
42 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
43 \ifdefined\PackageError
44 \def\loi_error#1{\PackageError\loiname{#1}{Read the manual}}% pour LaTeX
45 \else
46 \def\loi_error#1{\errmessage{Package \loiname\space Error: #1^^J}}% pour TeX
47 \fi
48
49 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% vérification de la présence de etex %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
51 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
52 \begingroup
53 \edef\__tempa{\meaning\eTeXversion}\edef\__tempb{\string\eTeXversion}%
54 \ifx\__tempa\__tempb
55 \endgroup
56 \else
57 \endgroup
58 \loi_error{You are not using an eTeX engine, listofitems cannot work.}%
59 \loi_restorecatcode\expandafter\endinput
60 \fi
61
62 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macros auxiliaires %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
64 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
65 \chardef\loi_stop=0
66 \def\loi_quark{\loi_quark}
67 \long\def\loi_identity#1{#1}
68 \long\def\loi_gobarg#1{}
69 \long\def\loi_first#1#2{#1}
70 \long\def\loi_second#1#2{#2}
71 \long\def\loi_firsttonil#1#2\_nil{#1}
72 \long\def\loi_antefi#1#2\fi{#2\fi#1}
73 \long\def\loi_exparg#1#2{\expandafter\loi_exparg_a\expandafter{#2}{#1}}% \loi_exparg{<a>}{<b>} ↵
74 \long\def\loi_exparg_a#1#2{#2{#1}}
75 \long\def\loi_expafter#1#2{\expandafter\loi_expafter_a\expandafter{#2}{#1}}% \loi_expafter{<a>}{<b>}
76 \long\def\loi_expafter_a#1#2{#2#1}
77 \def\loi_macroname{\loi_ifinrange\escapechar[[0:255]]{\expandafter\loi_gobarg}{}\string}
78 \def\loi_argcsname#1#\loi_argcsname_a{#1}
79 \def\loi_argcsname_a#1#2{\loi_expafter{#1}{\csname#2\endcsname}}
80 \long\def\loi_addtomacro#1#2{\loi_exparg{\def#1}{#1#2}}
81

```



```

82 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
83 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macros de test %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
84 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
85 \long\def\loi_ifnum#1{\ifnum#1\expandafter\loi_first\else\expandafter\loi_second\fi}
86 \long\def\loi_ifx#1{\ifx#1\expandafter\loi_first\else\expandafter\loi_second\fi}
87 \long\def\loi_ifempty#1{\loi_exparg\loi_ifx{\expandafter\relax\detokenize{#1}\relax}}
88 \def\loi_ifstar#1#2{\def\loi_ifstar_a{\loi_ifx{*}\loi_nxttok}{\loi_first{#1}}{#2}}\futurelet\
loi_nxttok\loi_ifstar_a}
89 \long\def\loi_ifprimitive#1{\edef\loi_tempa{\meaning#1}\edef\loi_tempb{\string#1}\loi_ifx{\
loi_tempa\loi_tempb}}
90 \long\def\loi_ifcs#1{% #1 est-il une sc (n'étant pas une primitive) ?
91 \loi_ifempty{#1}
92 {\loi_second
93 }
94 {\loi_ifspacefirst{#1}
95 \loi_second% si espace en 1er, faux
96 {\loi_exparg\loi_ifempty{\loi_gobarg#1}}% 1 seul token ?
97 {\begingroup \escapechar'\_
98 \if\expandafter\loi_firsttonil\detokenize{#1}\_nil\expandafter\loi_firsttonil\string\
relax\_nil
99 \loi_ifprimitive{#1}
100 {\endgroup\expandafter\loi_second}
101 {\endgroup\expandafter\loi_first}%
102 \else
103 \endgroup\expandafter\loi_second
104 \fi
105 }
106 {\loi_second% si plusieurs tokens, faux
107 }%
108 }%
109 }%
110 }
111 \def\loi_ifinrange#1[[#2:#3]]{\loi_ifnum{\numexpr(#1-#2)*(#1-#3)>0 }\loi_second\loi_first}
112 \def\loi_ifstring#1\in#2{% si la chaine #1 est contenue dans #2
113 \def\loi_ifstring_a##1#1##2\_nil{\loi_ifempty{##2}\loi_second\loi_first}%
114 \loi_ifstring_a#2#1@\nil% appel de la macro auxiliaire
115 }
116
117 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
118 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macros \loi_foreach %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
119 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
120 \newcount\loi_cnt_foreach_nest \loi_cnt_foreach_nest=0
121 \def\end_foreach{\end_foreach}
122 \def\loi_def_foreachsep#1{%
123 \long\def\loi_foreach##1\in##2##3{%
124 \global\advance\loi_cnt_foreach_nest1
125 \loi_argc\name\def{loop_code\_number\loi_cnt_foreach_nest}{##3}%
126 \loi_foreach_a##1##2#1\end_foreach#1%
127 \loi_argc\name\let{loop_code\_number\loi_cnt_foreach_nest}\empty
128 \global\advance\loi_cnt_foreach_nest-1
129 }%
130 \long\def\loi_foreach_a##1##2#1{%
131 \def##1{##2}%
132 \loi_ifx{\end_foreach##1}
133 {}
134 {\csname loop_code\_number\loi_cnt_foreach_nest\endcsname% exécute le code

```

```

135 \loi_foreach_a##1%
136 }%
137 }%
138 }
139
140 %%%%%%%%%%
141 %%%%%%%%%% macros gérant l'appariement %%%%%%%%%%
142 %%%%%%%%%%
143 \long\def\defpair#1{%
144 \let\loi_listofpair\empty
145 \loi_ifempty{#1}
146 {}
147 {\defpair_a{#1}\loi_quark\loi_quark}%
148 }
149 \long\def\defpair_a#1#2#3{%
150 \loi_ifx{\loi_quark#2}
151 {\def\loi_sanitizelist##1,\_nil{\def\loi_listofpair{##1}}%
152 \loi_sanitizelist#1\_nil
153 }
154 {\loi_if_validpair#2#3%
155 {\long\def\loi_paired_a{#2}\long\def\loi_paired_b{#3}%
156 \loi_ifx{\loi_paired_a\loi_paired_b}
157 {\loi_error{Paired tokens must not be equal, the pair \detokenize{#2#3} is ignored}%
158 \defpair_a{#1}%
159 }
160 {\defpair_a{#1#2#3,}%
161 }%
162 }
163 {\loi_error{Invalid paired tokens, the pair "\detokenize{#2}" and "\detokenize{#3}" is
164 ignored}%
165 \defpair_a{#1}%
166 }%
167 }
168 \long\def\loi_if_validpair#1#2{%
169 \def\loi_validpair{1}%
170 \loi_if_invalid_pairedtoken{#1}{\def\loi_validpair{0}}%
171 \loi_if_invalid_pairedtoken{#2}{\def\loi_validpair{0}}%
172 \loi_ifnum{\loi_validpair=1 }
173 }
174 \long\def\loi_if_invalid_pairedtoken#1{%
175 \loi_ifempty{#1}
176 {\loi_identity
177 }
178 {\loi_ifspacefirst{#1}
179 {\loi_identity
180 }
181 {\loi_exparg\loi_ifempty{\loi_gobarg#1}% 1 seul token ?
182 {\ifcat\relax\noexpand#1\expandafter\loi_identity\else\expandafter\loi_gobarg\fi}
183 {\loi_identity}% si plusieurs tokens, faux
184 }%
185 }%
186 }
187 \long\def\loi_count_occur#1\in#2:#3{% compte le nombre d'occurrences de #1 dans #2 et met le
188 résultat dans la macro #3
189 \long\def\loi_count_occur_a#1##2#1##3\_nil{%

```

```

189 \loi_ifempty{##3}
190   {\def#3{##1}}
191   {\expandafter\loi_count_occur_a\number\numexpr##1+1\relax##3\_nil}%
192 }%
193 \loi_count_occur_a0#2#1\_nil
194 }
195 \long\def\loi_check_pair#1#2\in#3{% teste l'appariement de #1 et #2 dans #3
196 \loi_ifempty{#3}
197   {\loi_second
198   }
199   {\loi_count_occur#1\in#3:\loi_tempa
200   \loi_count_occur#2\in#3:\loi_tempb
201   \loi_ifnum{\loi_tempa=\loi_tempb\relax}%
202   }%
203 }
204 \long\def\loi_grabpaired_expr#1#2#3#4#5{% #1=liste de paires #2=expression #3=séparateur #4=
résultat #5=ce qui reste
205 \let#4\empty
206 \def\loi_remain{#2#3}%
207 \loi_foreach\loi_pair\in{#1}{\expandafter\loi_grabpaired_expr_a\loi_pair{#3}#4}%
208 \def\loi_remove_lastsep##1#3\_nil{\def#4{##1}}%
209 \expandafter\loi_remove_lastsep#4\_nil
210 \expandafter\long\expandafter\def\expandafter\loi_grab_remain#4##1\_nil{\loi_ifempty{##1}{\let#5\
empty}{\loi_exparg{\def#5}{\loi_gobarg##1}}}%
211 \loi_grab_remain#2\_nil
212 }
213 \long\def\loi_grabpaired_expr_a#1#2#3#4{% #1#2=paire en cours #3=séparateur #4=résultat
214 \loi_exparg{\loi_check_pair#1#2\in}#4% si les paires sont appariées dans le résultat
215   {}% passer à la paire suivante
216   {\long\def\loi_grabpaired_expr_b##1#3##2\_nil{%
217     \loi_addtomacro#4{##1#3}% ajouter au résultat ce qui est jusqu'au prochain séparateur
218     \def\loi_remain{##2}%
219     \loi_exparg{\loi_check_pair#1#2\in}{#4}
220     {}
221     {\loi_ifempty{##2}
222     {\loi_error{"\detokenize{##1}" and "\detokenize{##2}" are not paired}}
223     {\loi_grabpaired_expr_b##2\_nil}%
224     }%
225     }%
226     \expandafter\loi_grabpaired_expr_b\loi_remain\_nil
227     }%
228   }
229 \def\insidepair#1#2#3#4{% #1#2=paire #3=expr #4=macro recevant le resultat
230 \loi_if_validpair#1#2%
231   {\loi_ifcs{#3}
232   {\loi_exparg{\insidepair#1#2}{#3}#4%
233   }
234   {\loi_check_pair#1#2\in{#3}% si les paires sont appariées dans le résultat
235   {\def\insidepair_a##1#1##2\_nil{\insidepair_b##2\_nil{##1}}%
236   \def\insidepair_b##1#2##2\_nil##3{%
237     \loi_check_pair#1#2\in{##3##1#2}
238     {\loi_exparg{\def#4}{\loi_gobarg##3##1}}%
239     {\insidepair_b##2\_nil{##3##1#2}}%
240     }%
241     \insidepair_a#3\_nil
242     }

```

```

243     {\loi_error{"\detokenize{#1}" and "\detokenize{#2}" are not paired in "#3"}%
244     }%
245   }%
246 }
247 {\loi_error{Invalid paired tokens "\detokenize{#1}" and "\detokenize{#2}", empty \string#4 ↵
248   returned}% et bim
249 \let#4\empty% voilà, bien fait pour vos gueules
250 }%
251 }
252 %%%
253 %%% macro \loi_fornum %%%
254 %%%
255 \def\loi_fornum#1=#2to#3\do{%
256   \edef#1{\number\numexpr#2}\edef\loi_sgncomp{\ifnum#1<\numexpr#3\relax>+\else<-\fi}%
257   \expandafter\loi_fornum_a\csname loi_fornum_\string#1\expandafter\endcsname\expandafter{\number\
258     numexpr#3\expandafter}\loi_sgncomp#1%
259 }
260 \long\def\loi_fornum_a#1#2#3#4#5#6{\def#1{\unless\ifnum#5#3#2\relax\loi_antefi{#6\edef#5{\number\
261   numexpr#5#41\relax}#1}\fi}#1}
262 %%%
263 %%% macro retirant les espaces extrêmes %%%
264 %%%
265 \long\def\loi_ifspacefirst#1{\expandafter\loi_ifspacefirst_a\detokenize{#10} \_nil}
266 \long\def\loi_ifspacefirst_a#1 #2\_nil{\loi_ifempty{#1}}
267 \expandafter\def\expandafter\loi_gobspace\space{}
268 \def\loi_removefirstspaces{\romannumeral\loi_removefirstspaces_a}
269 \long\def\loi_removefirstspaces_a#1{\loi_ifspacefirst{#1}{\expandafter\loi_removefirstspaces_a\
270   \expandafter{\loi_gobspace#1}}{\loi_stop#1}}
271 \edef\loi_restorezerocatcode{\catcode0=\number\catcode0 \relax}
272 \catcode0 12
273 \long\def\loi_removefirstspaces_a#1^^00^^00\_nil}
274 \long\def\loi_removefirstspaces_a#1 ^^00{\loi_removefirstspaces_b#1^^00}
275 \long\def\loi_removefirstspaces_b#1^^00#2\_nil{\loi_ifspacefirst{#2}{\loi_removefirstspaces_a#1^^00 ↵
276   ^^00\_nil}}{\loi_stop#1}}
277 \loi_restorezerocatcode
278 \long\def\loi_removeextremespaces#1{% #1=texte où les espaces extrêmes sont retirés
279 \romannumeral\expandafter\expandafter\expandafter\loi_removefirstspaces\expandafter\expandafter\
280   \expandafter\expandafter\expandafter\loi_stop\loi_removefirstspaces{#1}}%
281 %%%
282 %%% macro publique \setsepchar %%%
283 %%%
284 \def\setsepchar{\futurelet\loi_nxttok\setsepchar_a}
285 \def\setsepchar_a{\loi_ifx{[\loi_nxttok]\setsepchar_b{\setsepchar_b[/]}}
286 \long\def\setsepchar_b[#1]#2{% #1=sepcar de <liste des sepcar> #2=<liste des sepcar>
287 \loi_ifempty{#1}
288 {\loi_error{Empty separator not allowed, separator "/" used}%
289 \setsepchar_b[/]{#2}%
290 }
291 {\def\loi_currentsep{#1}%
292 \removeextremespacesfalse
293 \loi_nestcnt1 % réinitialiser niveau initial à 1

```

```

293 \def\nestdepth{1}%
294 \loi_argcname\let{loi_previndex[\number\loi_nestcnt]}\empty
295 \def\loi_listname{loi_listofsep}%
296 \let\loi_def\def \let\loi_edef\edef \let\loi_let\let
297 \let\loi_listofpair_saved\loi_list_ofpair
298 \let\loi_list_ofpair\empty
299 \loi_ifempty{#2}
300 {\loi_error{Empty list of separators not allowed, "," used}%
301 \readlist_e1{,}%
302 }
303 {\readlist_e1{#2}%
304 }%
305 \loi_argcname\let\nestdepth{loi_listofseplen[0]}%
306 \loi_argcname\let\loi_currentsep{loi_listofsep[1]}% 1er car de séparation
307 \let\loi_listofpair\loi_listofpair_saved
308 }%
309 }
310
311 %%%%%%%%%%%
312 %%%%%%%%%%% macro normalisant l'index %%%%%%%%%%%
313 %%%%%%%%%%%
314 \def\loi_normalizeindex#1#2#3{% #1=macroname #2=liste d'index #3=profondeur max --> renvoie {err↵
    }{indx norm}
315 \loi_ifempty{#2}
316 {\loi_stop{}{}}
317 {\loi_normalizeindex_a1}{#3}{#1}#2,\loi_quark,}%
318 }%
319 \def\loi_normalizeindex_a#1#2#3#4#5,{% #1=compteur de profondeur #2=index précédents #3=profondeur ↵
    max #4=macroname #5=index courant
320 \loi_ifx{\loi_quark#5}
321 {\loi_normalizeindex_c#2\loi_quark% supprimer la dernière virgule
322 }
323 {\loi_ifnum{#1>#3 }
324 {\loi_invalidindex{Too deeply nested index, index [.] retained}{#2}% si profondeur trop ↵
    grande
325 }
326 {\loi_ifinrange\ifnum\numexpr#5<0 -1*\fi(#5)[[1:\csname #4len[#20]\endcsname]]% si abs(#5) ↵
    hors de [1,len]
327 {\loi_exparg\loi_normalizeindex_b{\number\numexpr#5\ifnum\numexpr#5<0 +\csname #4len[#20]\↵
    endcsname+1\fi}{#1}{#2}{#3}{#4}}
328 {\loi_invalidindex{#5 is an invalid index, index [.] retained}{#2}}%
329 }%
330 }%
331 }
332 \def\loi_normalizeindex_b#1#2#3{\loi_exparg\loi_normalizeindex_a{\number\numexpr#2+1}{#3#1,}}% #1=↵
    index à rajouter #2=compteur de profondeur #3=index précédents
333 \def\loi_normalizeindex_c#1,\loi_quark{\loi_stop{}{#1}}
334 \def\loi_invalidindex#1#2{\loi_ifempty{#2}{\loi_invalidindex_a{#1},}\loi_invalidindex_a{#1}{#2}}
335 \def\loi_invalidindex_a#1#2{\loi_invalidindex_b#1\loi_quark#2\loi_quark}
336 \def\loi_invalidindex_b#1[.]#2\loi_quark#3,\loi_quark#4\loi_quark,{\loi_stop{#1[#3]#2}{#3}}% #4= ↵
    index ignorés
337
338 %%%%%%%%%%%
339 %%%%%%%%%%% macro publique \readlist %%%%%%%%%%%
340 %%%%%%%%%%%
341 \newcount\loi_nestcnt

```

```

342 \def\readlist{\let\loi_def\gdef\let\loi_edef\xdef\def\loi_let{\global\let}\readlist_a}%
343 \def\readlist{\let\loi_def\def\let\loi_edef\edef\let\loi_let\let\readlist_a}
344 \def\readlist_a{%
345   \loi_nestcnt1 % niveau initial = 1
346   \loi_argcsname\let\loi_previndex[\number\loi_nestcnt]}empty
347   \loi_ifstar{\_removeextremespacestrue\readlist_b}{\_removeextremespacesfalse\readlist_b}%
348 }
349 \long\def\readlist_b#1#2{% #1=macro stockant les éléments #2=liste des éléments
350   \loi_ifcs{#2}
351     {\loi_exparg{\readlist_b#1}{#2}%
352     }
353     {\loi_edef\loi_listname{\loi_macroname#1}%
354     \loi_argcsname\loi_let{\loi_listname nest}\nestdepth
355     \loi_argcsname\loi_def{\loi_listname[]}{#2}% la liste entière
356     \loi_argcsname\loi_def{\loi_listname sep[]}{ }% séparateur vide
357     \loi_ifempty{#2}
358       {\loi_def#1[##1]}%
359       \loi_argcsname\loi_def{\loi_listname len}{0}\loi_argcsname\loi_def{\loi_listname len[0]}{0}%
360       \loi_error{Empty list ignored, nothing to do}%
361     }
362     {\loi_edef#1[##1]{\unexpanded{\romannumeral\expandafter\loi_checkindex\romannumeral\
loi_normalizeindex}{\loi_listname}{##1}{\csname\loi_listname nest\endcsname}{\
loi_listname}}%
363     \loi_argcsname\loi_edef{\loi_listname sep}{##1}{\unexpanded{\romannumeral\expandafter\
loi_checkindex\romannumeral\loi_normalizeindex}{\loi_listname}{##1}{\csname\loi_listname
nest\endcsname}{\loi_listname sep}}%
364     \readlist_c{#2}%
365     \loi_argcsname\loi_argcsname\loi_let{\loi_listname len}{\loi_listname len[0]}% longueur du
niveau 0
366   }%
367 }%
368 }
369 \def\loi_checkindex#1#2#3{%
370   \expandafter\expandafter\expandafter\loi_stop\csname#3[#2]\expandafter\endcsname
371   \romannumeral\loi_ifempty{#1}{\loi_stop}{\loi_stop\loi_error{#1}}%
372 }
373 \def\readlist_c{%
374   \loi_argcsname\loi_let\loi_currentsep{\loi_listofsep[\number\loi_nestcnt]}%
375   \expandafter\readlist_d\loi_currentsep|\_nil
376 }
377 \long\def\readlist_d#1||#2\_nil#3{\readlist_e1{#3#1}}% #1=<sep courant simple> #3=liste -> rajoute
un élément vide pour le test \ifempty ci dessus
378 \long\def\readlist_e#1#2{% #1=compteur d'index #2=liste d'éléments à examiner terminée par <sep
courant simple> >>RIEN laissé après
379   \loi_ifempty{#2}
380     {\loi_argcsname\loi_edef{\loi_listname len[\csname\loi_previndex[\number\loi_nestcnt]\endcsname
0]}{\number\numexpr#1-1\relax}%
381     \loi_argcsname\loi_let{\loi_listname sep[\csname\loi_previndex[\number\loi_nestcnt]\endcsname\
number\numexpr#1-1\relax]}empty% le dernier <sep> est <vide> ##NEW v1.52
382     \advance\loi_nestcnt-1
383     \loi_argcsname\loi_let\loi_currentsep{\loi_listofsep[\number\loi_nestcnt]}%
384   }
385   {\loi_expafter{\readlist_f{#2}{}}\loi_currentsep|\_loi_quark||#2\_nil{#1}}% aller isoler le 1er
item
386 }%
387 }

```

```

388 \long\def\readlist_f#1#2#3||{% #1=liste restante #2=<dernier sep utilisé> #3=<sep courant>
389 \loi_ifx{\loi_quark#3}% on a épuisé tous les <séparateurs> ? RESTE à lire <expr+sep1>\_nil{<
compteur>}
390 {\loi_ifempty{#2}% si #2 vide, aucun <sep utilisé> n'a été trouvé, il reste à lire "<liste ✓
complète>\_nil"
391 {\long\def\readlist_g##1\_nil##2{\loi_exparg{\readlist_h{##2}{}}{\loi_gobarg##1}{#2}}% ##2=✓
compteur d'index
392 }
393 {\loi_ifx{\loi_listofpair\empty}% paires définies ?
394 {\long\def\readlist_g##1#2##2\_nil##3{\loi_exparg{\readlist_h{##3}{##2}}{\loi_gobarg ✓
##1}{#2}}%
395 }
396 {\long\def\readlist_g##1\_nil##2{%
397 \loi_exparg{\loi_exparg\loi_grabpaired_expr\loi_listofpair}{\loi_gobarg##1}{#2}\ ✓
loi_grabpaired_result\loi_grabpaired_remain
398 \loi_exparg{\loi_exparg{\readlist_h{##2}}{\loi_grabpaired_remain}}{\loi_grabpaired_result ✓
}{#2}}%
399 }%
400 }%
401 \readlist_g\relax% le \relax meuble l'argument délimité
402 }
403 {\long\def\readlist_g##1#3##2\_nil{%
404 \loi_ifempty{##2}% si <liste restante> ne contient pas le <sep courant>
405 {\readlist_f{#1}{#2}% recommencer avec le même <sep utile>
406 }%
407 {\loi_ifx{\loi_listofpair\empty}% si pas de paires définies
408 {\loi_exparg\readlist_f{\loi_gobarg##1#3}{#3}% raccourcir <liste restante> et <sep ✓
courant>:=<sep utile>% ##BUGFIX v1.53
409 }%
410 {\loi_exparg\loi_grabpaired_expr\loi_listofpair{#1}{#3}\loi_grabpaired_result\ ✓
loi_grabpaired_remain
411 \loi_exparg\readlist_f{\loi_grabpaired_result#3}{#3}%
412 }%
413 }%
414 }%
415 \readlist_g\relax#1#3\_nil% ##BUGFIX v1.53
416 }%
417 }
418 \long\def\readlist_h#1#2#3{% #1=compteur d'index #2=liste restante #3=élément courant
419 \loi_ifnum{0\loi_exparg\loi_ifspacefirst{\loi_currentsep}{1}\if_removeextremespaces1\fi=11 }% s' ✓
il faut retirer les espaces extrêmes
420 {\loi_exparg{\loi_exparg{\readlist_i{#1}{#2}}{\loi_removeextremespaces{#3}}}% redéfinir l' ✓
élément courant
421 {\readlist_i{#1}{#2}{#3}}%
422 }
423 \long\def\readlist_i#1#2#3#4{% #1=compteur d'index #2=liste restante #3=élément courant #4=sep ✓
utilisé
424 \loi_ifnum{0\if_ignoreemptyitems1\fi\loi_ifempty{#3}1{=11 }
425 {\readlist_e{#1}{#2}% si l'on n'ignore pas les éléments vides
426 }%
427 {\loi_argcname\loi_def{\loi_listname[\csname loi_previndex[\number\loi_nestcnt]\endcsname ✓
#1]}{#3}% assignation de l'item ctuel à la macro
428 \loi_argcname\loi_def{\loi_listname sep[\csname loi_previndex[\number\loi_nestcnt]\endcsname ✓
#1]}{#4}% assignation du <sep> actuel à la macro \<macrolist>sep
429 \loi_ifnum{\loi_nestcnt<nestdepth\relax}% si imbrication max non atteinte
430 {\advance\loi_nestcnt1

```

```

431 \loi_argcsname\edef\loi_previndex[\number\loi_nestcnt]]{\csname loi_previndex[\number\numexpr\
\loi_nestcnt-1]\endcsname#1,}%
432 \readlist_c{#3}% recommencer avec l'élément courant
433 }
434 {}%
435 \loi_exparg\readlist_e{\number\numexpr#1+1}{#2}% puis chercher l'élément suivant dans la liste ↵
restante
436 }%
437 }
438 }
439 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
440 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macro \listlen %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
441 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
442 \def\listlen#1[#2]{%
443 \romannumeral\loi_ifempty{#2}
444 {\expandafter\expandafter\expandafter\loi_stop\csname\loi_macroname#1len[0]\endcsname}
445 {\loi_exparg\listlen_a{\romannumeral-'\.loi_macroname#1}{#2}}%
446 }
447 \def\listlen_a#1#2{% #1=macro name #2=index non normalisé prendre <profondeur max-1>
448 \loi_exparg{\expandafter\listlen_b\romannumeral\loi_normalizeindex{#1}{#2}}{\number\numexpr\
\csname#1nest\endcsname-1}{#1}%
449 }
450 \def\listlen_b#1#2#3{% #1=err #2=index normalisé #3=macroname
451 \expandafter\expandafter\expandafter\loi_stop\csname#3len[#2,0]\expandafter\endcsname
452 \romannumeral\loi_ifempty{#1}{\loi_stop}{\loi_stop\loi_error{#1}}%
453 }
454 }
455 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
456 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macro \foreachitem %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
457 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
458 \def\foreachitem#1\in#2{%
459 \edef\foreachitem_a{\noexpand\foreachitem_c\noexpand#1{\expandafter\noexpand\csname\loi_macroname\
#1cnt\endcsname}{\loi_macroname#2}}%
460 \futurelet\loi_nxttok\foreachitem_b
461 }
462 \def\foreachitem_b{\loi_ifx{\loi_nxttok[]\foreachitem_a{\foreachitem_a[]}}
463 \def\foreachitem_c#1#2#3[#4]{% prendre <profondeur max-1>
464 \loi_exparg{\expandafter\foreachitem_d\romannumeral\loi_normalizeindex{#3}{#4}}{\number\numexpr\
\csname#3nest\endcsname-1}{#1}{#2}{#3}%
465 }
466 \def\foreachitem_d#1#2{\loi_ifempty{#2}{\foreachitem_e{#1}{}}{\foreachitem_e{#1}{#2,}}% #1=err ↵
#2=index norm
467 \long\def\foreachitem_e#1#2#3#4#5#6{% #1=err #2=index norm #3=macroiter #4=compteur associé #5=
nom de macrolist #6=code
468 \loi_ifnum{\csname#5len[#20]\endcsname>0 }
469 {\loi_ifempty{#1}{\loi_error{#1}}%
470 \loi_fornum#4=1to\csname#5len[#20]\endcsname\do{\loi_argcsname\let#3{#5[#2#4]}#6}%
471 }
472 {}%
473 }
474 }
475 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
476 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macro \showitem %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
477 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
478 \def\showitems{\loi_ifstar{\let\showitems_cmd\detokenize\showitems_a}{\let\showitems_cmd\
loi_identity\showitems_a}}

```



```

479 \def\showitems_a#1{\def\showitems_b{\showitems_d#1}\futurelet\loi_nxttok\showitems_c}
480 \def\showitems_c{\loi_ifx{\loi_nxttok[]}\showitems_b{\showitems_b[]}}
481 \def\showitems_d#1[#2]{\foreachitem\showitems_ater\in#1[#2]{\showitemsmacro{\expandafter\
showitems_cmd\expandafter{\showitems_ater}}}}
482 \unless\ifdefined\fbbox
483 \newdimen\fbboxrule \newdimen\fbboxsep \fbboxrule=.4pt \fbboxsep=3pt % réglages identiques à LaTeX
484 \def\fbbox#1{% imitation de la macro \fbbox de LaTeX, voir pages 271 à 274 de "Apprendre à
programmer en TeX"
485 \hbox{%
486 \vrule width\fbboxrule
487 \vtop{%
488 \vbox{\hrule height\fbboxrule \kern\fbboxsep \hbox{\kern\fbboxsep#1\kern\fbboxsep}}%
489 \kern\fbboxsep \hrule height\fbboxrule
490 } \vrule width\fbboxrule
491 }%
492 }
493 \fi
494 \def\showitemsmacro#1{% encadrement par défaut
495 \begingroup\fbboxsep=0.25pt \fbboxrule=0.5pt \fbbox{\strut#1}\endgroup
496 \hskip0.25em\relax
497 }
498
499 %%%
500 %%% macro \itemtomacro %%%
501 %%%
502 \def\itemtomacro#1[#2]{% #1[#2]=item non encore lu: #3=macro
503 \edef\loi_listname{\loi_macroname#1}%
504 \loi_exparg{\expandafter\itemtomacro_a\romannumeral\expandafter\loi_normalizeindex\expandafter{\
loi_listname}{#2}}{\csname\loi_listname nest\endcsname}\let
505 }
506 \def\gitomacro#1[#2]{% #1[#2]=item
507 \xdef\loi_listname{\loi_macroname#1}%
508 \loi_exparg{\expandafter\itemtomacro_a\romannumeral\expandafter\loi_normalizeindex\expandafter{\
loi_listname}{#2}}{\csname\loi_listname nest\endcsname}{\global\let}%
509 }
510 \def\itemtomacro_a#1#2#3#4{%
511 \loi_ifempty{#1}{\loi_error{#1}}%
512 \loi_argc\name#3#4{\loi_listname[#2]}%
513 }
514
515 %%%
516 %%% réglages par défaut %%%
517 %%%
518 \newif\if_removeextremespaces
519 \newif\if_ignoreemptyitems
520 \let\ignoreemptyitems\_ignoreemptyitemstrue
521 \let\reademptyitems\_ignoreemptyitemsfalse
522 \setsepchar{,}
523 \defpair{}
524 \loi_def_foreachsep{,}
525 \reademptyitems
526
527 \loi_restorecatcode
528 \endinput
529
530 #####

```

```

531 ##### Historique #####
532 #####
533
534 v1.0 19/8/2016
535 - Première version publique
536
537 v1.1 01/09/2016
538 - Stockage des séparateurs dans <macrolist>sep
539 - bug corrigé dans \loi_restorecatcode
540
541 v1.2 22/10/2016
542 - macros \greadlist et \gitemtomacro pour la globalité
543
544 v1.3 18/11/2016
545 - bugs corrigés dans la gestion de la globalité
546
547 v1.4 05/10/2017
548 - test \loi_ifprimitive ajouté au test \loi_ifcs
549 - suppression de \loi_expafternil, création de \loi_expafter,
550 modification de \loi_argcsname
551 - correction d'un bug : \setsepchar{\par} ne provoque plus
552 d'erreur. \loi_ifnum devient \long
553
554 v1.5 06/10/2017
555 - correction d'un bug dans \loi_ifcs
556
557 v1.51 24/10/2017
558 - correction d'un bug dans \loi_ifcs
559
560 v1.52 13/01/2018
561 - le dernier séparateur est <vide>
562
563 v1.53 13/03/2018
564 - correction d'un bug dans \readlist_g
565
566 v1.6 01/11/2018
567 - possibilité d'appariement de tokens dans les items

```