Package

listofitems

v1.61

3 March 2019

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)LaTeX 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]` is[3] "`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 ⟨*list*⟩ 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 ⟨*list*⟩.
The macro `\foosep` is created. It is used with the syntax `\foosep[`⟨*index*⟩`]` and allows access to the parsing-separator that follows the item of rank ⟨*index*⟩. The last parsing-separator (the one following the last item) is empty. If the ⟨*index*⟩ 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])
```

1) 17 (parsing separator = -)
2) 8 (parsing separator = +)
3) 4 (parsing separator = -)
4) 11 (parsing separator =)

Number of items If we write `\readlist`⟨*macro-list*⟩`{`⟨*list*⟩`}`, then the macro ⟨*macro-list*⟩`len` contains[4] the number of the items in ⟨*list*⟩. In the example with `\foo`, the macro `\foolen` expands to 8.

View all items For purposes of debugging, the macro `\showitems`⟨*macro-list*⟩ includes all items from a list, while the star version displays these items "detokenized." [5]

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

12 abc x y z ‖ TEX ‖ !
12 abc x y {\bfseries z} ‖ \TeX ‖ !

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

```
\newcommand\showitemsmacro[1]{%
    \begingroup\fboxsep=0.25pt \fboxrule=0.5pt \fbox{\strut#1}\endgroup
    \hskip0.25em\relax}
```

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

The macro `\fbox` and associated dimensions `\fboxsep` and `\fboxrule`, 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 ⟨*list*⟩, execute the starred version `\readlist*`⟨*macro*⟩`{`⟨*list*⟩`}`:

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

12 abc x y z ‖ TEX ‖ !

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`.

[4] That is to say, it is purely expandable and grows into a number.
[5] The 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:

```
\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
```

a) number of items = 7
[12] [abc] [x y] [**z**] [ ] [TEX] [!]
b) number of items = 6
[12] [abc] [x y] [**z**] [TEX] [!]

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

```
\setsepchar{ }% parsing-separator = space
\readlist\phrase{One phrase to test.}
\foreachitem\word\in\phrase{List item number \wordcnt{}:  \word\par}
```

List item number 1: One
List item number 2: phrase
List item number 3: to
List item number 4: test.

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

```
\setsepchar{ }% parsing-separator = space
\readlist\phrase{One phrase to test.}
\itemtomacro\phrase[2]\aword
\meaning\aword\par
\itemtomacro\phrase[-1]\wordattheend
\meaning\wordattheend
```

macro:->phrase
macro:->test.

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[⟨*delimiter*⟩]{⟨*delimited-list-of-parsing-separators*⟩}.
By default, the ⟨*delimiter*⟩ 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 ⟨*depth*⟩ 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:

```
\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]
```

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

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

```
\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])
```

Term 1: 1
Term 2: 2*3 (factors: 2 and 3)
Term 3: 4/5*6 (factors: 4, 5 and 6)

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.

```
\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]
```

a) 3 or 3
b) 3
c) 3
d) 5
e) 1
f) 2
g) 3

Displaying list items The macro `\showitems`⟨*macrolist*⟩`[`⟨*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*⟩.

```
\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]
```

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]

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.

```
\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
```

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]

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*⟩`]`.

```
\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
```

2nd verse = Who could speed even faster than light.
A word = even

The macro `\gitemtomacro` 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.

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

[1] [2*[3+4*(5+6-7)+8]] [9] [10]

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:

```
% !TeX encoding = ISO-8859-1
% Ce fichier contient le code de l'extension "listofitems"
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                     %
\def\loiname                 {listofitems}                            %
\def\loiver                      {1.61}                               %
%                                                                     %
\def\loidate                  {2019/03/03}                            %
%                                                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                     %
% Author     : Christian Tellechea, Steven B. Segletes                %
% Status     : Maintained                                             %
% Maintainer : Christian Tellechea                                    %
% Email      : unbonpetit@netc.fr                                     %
%              steven.b.segletes.civ@mail.mil                         %
% Package URL: https://www.ctan.org/pkg/listofitems                   %
% Bug tracker: https://framagit.org/unbonpetit/listofitems/issues     %
% Repository : https://framagit.org/unbonpetit/listofitems/tree/master
% Copyright  : Christian Tellechea 2016-2019                          %
% Licence    : Released under the LaTeX Project Public License v1.3c %
%              or later, see http://www.latex-project.org/lppl.txt    %
% Files      : 1) listofitems.tex                                     %
%              2) listofitems.sty                                     %
%              3) listofitems-fr.tex                                  %
%              4) listofitems-fr.pdf                                  %
```

```
%              5) listofitems-en.tex                                    %
%              6) listofitems-en.pdf                                    %
%              7) README                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\ifdefined\ProvidesPackage\else
  \immediate\write -1 {%
  Package: \loidate\space v\loiver\space Grab items in lists using user-specified sep char (CT)}%
\fi

\expandafter\edef\csname loi_restorecatcode\endcsname{\catcode\number`\_=\number\catcode`\_\relax}
\catcode`\_11

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% gestion des erreurs %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\ifdefined\PackageError
  \def\loi_error#1{\PackageError\loiname{#1}{Read the manual}}% pour LaTeX
\else
  \def\loi_error#1{\errmessage{Package \loiname\space Error: #1^^J}}% pour TeX
\fi

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%% vérification de la présence de etex %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\begingroup
\edef\__tempa{\meaning\eTeXversion}\edef\__tempb{\string\eTeXversion}%
\ifx\__tempa\__tempb
  \endgroup
\else
  \endgroup
  \loi_error{You are not using an eTeX engine, listofitems cannot work.}%
  \loi_restorecatcode\expandafter\endinput
\fi

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% macros auxiliaires %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\chardef\loi_stop=0
\def\loi_quark{\loi_quark}
\long\def\loi_identity#1{#1}
\long\def\loi_gobarg#1{}
\long\def\loi_first#1#2{#1}
\long\def\loi_second#1#2{#2}
\long\def\loi_firsttonil#1#2\_nil{#1}
\long\def\loi_antefi#1#2\fi{#2\fi#1}
\long\def\loi_exparg#1#2{\expandafter\loi_exparg_a\expandafter{#2}{#1}}% \loi_exparg{<a>}{<b>} devient <a>{<*b>}
\long\def\loi_exparg_a#1#2{#2{#1}}
\long\def\loi_expafter#1#2{\expandafter\loi_expafter_a\expandafter{#2}{#1}}% \loi_expafter{<a>}{<b>} devient <a><*b>
\long\def\loi_expafter_a#1#2{#2#1}
\def\loi_macroname{\loi_ifinrange\escapechar[[0:255]]{\expandafter\loi_gobarg}{}\string}
\def\loi_argcsname#1#{\loi_argcsname_a{#1}}
\def\loi_argcsname_a#1#2{\loi_expafter{#1}{\csname#2\endcsname}}
\long\def\loi_addtomacro#1#2{\loi_exparg{\def#1}{#1#2}}

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% macros de test %%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\long\def\loi_ifnum#1{\ifnum#1\expandafter\loi_first\else\expandafter\loi_second\fi}
\long\def\loi_ifx#1{\ifx#1\expandafter\loi_first\else\expandafter\loi_second\fi}
\long\def\loi_ifempty#1{\loi_exparg\loi_ifx{\expandafter\relax\detokenize{#1}\relax}}
\def\loi_ifstar#1#2{\def\loi_ifstar_a{\loi_ifx{*\loi_nxttok}{\loi_first{#1}}{#2}}\futurelet\↙
    loi_nxttok\loi_ifstar_a}
\long\def\loi_ifstuffexpandable#1{\def\loi_tempa{#1}\loi_exparg{\def\loi_tempb}{#1}\expandafter\↙
    unless\loi_ifx{\loi_tempa\loi_tempb}}
\long\def\loi_ifcsexpandable#1{% #1 est-il constitué d'une sc _développable_ ?
  \loi_ifempty{#1}
    {\loi_second
    }
    {\loi_ifspacefirst{#1}
      {\loi_second% si espace en 1er, faux
      }
      {%
      \loi_exparg\loi_ifempty{\loi_gobarg#1}% 1 seul token ?
        {\begingroup\escapechar`\_\def\_{#1}\expandafter\endgroup
          \csname loi_\if\expandafter\expandafter\expandafter\loi_firsttonil\expandafter\string\_\↙
              _nil\string _first\else second\fi\endcsname
            {\loi_ifstuffexpandable{#1}}
            {\loi_second}%
        }
        {\loi_second% si plusieurs tokens, faux
        }%
      }%
    }%
}
\def\loi_ifinrange#1[[#2:#3]]{\loi_ifnum{\numexpr(#1-#2)*(#1-#3)>0 }\loi_second\loi_first}
\def\loi_ifstring#1\in#2{% si la chaine #1 est contenue dans #2
  \def\loi_ifstring_a##1#1##2\_nil{\loi_ifempty{##2}\loi_second\loi_first}%
  \loi_ifstring_a#2#1\@nil% appel de la macro auxiliaire
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% macros \loi_foreach %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcount\loi_cnt_foreach_nest \loi_cnt_foreach_nest=0
\def\end_foreach{\end_foreach}
\def\loi_def_foreachsep#1{%
  \long\def\loi_foreach##1\in##2##3{%
    \global\advance\loi_cnt_foreach_nest1
    \loi_argcsname\def{loop_code_\number\loi_cnt_foreach_nest}{##3}%
    \loi_foreach_a##1##2#1\end_foreach#1%
    \loi_argcsname\let{loop_code_\number\loi_cnt_foreach_nest}\empty
    \global\advance\loi_cnt_foreach_nest-1
  }%
  \long\def\loi_foreach_a##1##2#1{%
    \def##1{##2}%
    \loi_ifx{\end_foreach##1}
      {}
      {\csname loop_code_\number\loi_cnt_foreach_nest\endcsname% exécute le code
      \loi_foreach_a##1%
      }%
```

```
  }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% macros gérant l'appariement %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\long\def\defpair#1{%
  \let\loi_listofpair\empty
  \loi_ifempty{#1}
    {}
    {\defpair_a{}#1\loi_quark\loi_quark}%
}
\long\def\defpair_a#1#2#3{%
  \loi_ifx{\loi_quark#2}
    {\def\loi_sanitizelist##1,\_nil{\def\loi_listofpair{##1}}%
    \loi_sanitizelist#1\_nil
    }
    {\loi_if_validpair#2#3%
      {\long\def\loi_paired_a{#2}\long\def\loi_paired_b{#3}%
      \loi_ifx{\loi_paired_a\loi_paired_b}
        {\loi_error{Paired tokens must not be equal, the pair \detokenize{#2#3} is ignored}%
        \defpair_a{#1}%
        }
        {\defpair_a{#1#2#3,}%
        }%
      }
      {\loi_error{Invalid paired tokens, the pair "\detokenize{#2}" and "\detokenize{#3}" is ↙
          ignored}%
      \defpair_a{#1}%
      }%
    }%
}
\long\def\loi_if_validpair#1#2{%
  \def\loi_validpair{1}%
  \loi_if_invalid_pairtoken{#1}{\def\loi_validpair{0}}%
  \loi_if_invalid_pairtoken{#2}{\def\loi_validpair{0}}%
  \loi_ifnum{\loi_validpair=1 }
}
\long\def\loi_if_invalid_pairtoken#1{%
  \loi_ifempty{#1}
    {\loi_identity
    }
    {\loi_ifspacefirst{#1}
      {\loi_identity
      }
      {\loi_exparg\loi_ifempty{\loi_gobarg#1}% 1 seul token ?
        {\ifcat\relax\noexpand#1\expandafter\loi_identity\else\expandafter\loi_gobarg\fi}
        {\loi_identity}% si plusieurs tokens, faux
      }%
    }%
}
\long\def\loi_count_occur#1\in#2:#3{% compte le nombre d'occurrences de #1 dans #2 et met le ↙
    résultat dans la macro #3
  \long\def\loi_count_occur_a##1##2#1##3\_nil{%
    \loi_ifempty{##3}
      {\def#3{##1}}
```

```
      {\expandafter\loi_count_occur_a\number\numexpr##1+1\relax##3\_nil}%
  }%
  \loi_count_occur_a0#2#1\_nil
}
\long\def\loi_check_pair#1#2\in#3{% teste l'appariement de #1 et #2 dans #3
  \loi_ifempty{#3}
    {\loi_second
    }
    {\loi_count_occur#1\in#3:\loi_tempa
    \loi_count_occur#2\in#3:\loi_tempb
    \loi_ifnum{\loi_tempa=\loi_tempb\relax}%
    }%
}
\long\def\loi_grabpaired_expr#1#2#3#4#5{% #1=liste de paires  #2=expression  #3=séparateur   #4=↙
    résultat    #5=ce qui reste
  \let#4\empty
  \def\loi_remain{#2#3}%
  \loi_foreach\loi_pair\in{#1}{\expandafter\loi_grabpaired_expr_a\loi_pair{#3}#4}%
  \def\loi_remove_lastsep##1#3\_nil{\def#4{##1}}%
  \expandafter\loi_remove_lastsep#4\_nil
  \expandafter\long\expandafter\def\expandafter\loi_grab_remain#4##1\_nil{\loi_ifempty{##1}{\let#5\↙
      empty}{\loi_exparg{\def#5}{\loi_gobarg##1}}}%
  \loi_grab_remain#2\_nil
}
\long\def\loi_grabpaired_expr_a#1#2#3#4{% #1#2=paire en cours  #3=séparateur   #4=résultat
  \loi_exparg{\loi_check_pair#1#2\in}#4% si les paires sont appariées dans le résultat
    {}% passer à la paire suivante
    {\long\def\loi_grabpaired_expr_b##1#3##2\_nil{%
      \loi_addtomacro#4{##1#3}% ajouter au résultat ce qui est jusqu'au prochain séparateur
      \def\loi_remain{##2}%
      \loi_exparg{\loi_check_pair#1#2\in}{#4}
        {}
        {\loi_ifempty{##2}
          {\loi_error{"\detokenize{#1}" and "\detokenize{#2}" are not paired}}
          {\loi_grabpaired_expr_b##2\_nil}%
        }%
      }%
    \expandafter\loi_grabpaired_expr_b\loi_remain\_nil
    }%
}
\def\insidepair#1#2#3#4{% #1#2=paire  #3=expr  #4=macro reçevant le resultat
  \loi_if_validpair#1#2%
    {\loi_ifcsexpandable{#3}
      {\loi_exparg{\insidepair#1#2}{#3}#4%
      }
      {\loi_check_pair#1#2\in{#3}% si les paires sont appariées dans le résultat
        {\def\insidepair_a##1#1##2\_nil{\insidepair_b##2\_nil{#1}}%
        \def\insidepair_b##1#2##2\_nil##3{%
          \loi_check_pair#1#2\in{##3##1#2}
            {\loi_exparg{\def#4}{\loi_gobarg##3##1}}%
            {\insidepair_b##2\_nil{##3##1#2}}%
          }%
        \insidepair_a#3\_nil
        }
        {\loi_error{"\detokenize{#1}" and "\detokenize{#2}" are not paired in "#3"}%
        }%
```

```
      }%
    }
    {\loi_error{Invalid paired tokens "\detokenize{#1}" and "\detokenize{#2}", empty \string#4 returned}% et bim
    \let#4\empty% voilà, bien fait pour vos gueules
    }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% macro \loi_fornum %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\loi_fornum#1=#2to#3\do{%
  \edef#1{\number\numexpr#2}\edef\loi_sgncmp{\ifnum#1<\numexpr#3\relax>+\else<-\fi}%
  \expandafter\loi_fornum_a\csname loi_fornum_\string#1\expandafter\endcsname\expandafter{\number\numexpr#3\expandafter}\loi_sgncmp#1%
}
\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\numexpr#5#41\relax}#1}\fi}#1}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% macro retirant les espaces extrêmes %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\long\def\loi_ifspacefirst#1{\expandafter\loi_ifspacefirst_a\detokenize{#10} \_nil}
\long\def\loi_ifspacefirst_a#1 #2\_nil{\loi_ifempty{#1}}
\expandafter\def\expandafter\loi_gobspace\space{}
\def\loi_removefirstspaces{\romannumeral\loi_removefirstspaces_a}
\long\def\loi_removefirstspaces_a#1{\loi_ifspacefirst{#1}{\expandafter\loi_removefirstspaces_a\expandafter{\loi_gobspace#1}}{\loi_stop#1}}
\edef\loi_restorezerocatcode{\catcode0=\number\catcode0 \relax}
\catcode0 12
\long\def\loi_removelastspaces#1{\romannumeral\loi_removelastspaces_a#1^^00 ^^00\_nil}
\long\def\loi_removelastspaces_a#1 ^^00{\loi_removelastspaces_b#1^^00}
\long\def\loi_removelastspaces_b#1^^00#2\_nil{\loi_ifspacefirst{#2}{\loi_removelastspaces_a#1^^00 ^^00\_nil}{\loi_stop#1}}
\loi_restorezerocatcode
\long\def\loi_removeextremespaces#1{% #1=texte où les espaces extrêmes sont retirés
  \romannumeral\expandafter\expandafter\expandafter\loi_removelastspaces\expandafter\expandafter\expandafter
  {\expandafter\expandafter\expandafter\loi_stop\loi_removefirstspaces{#1}}%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% macro publique \setsepchar %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\setsepchar{\futurelet\loi_nxttok\setsepchar_a}
\def\setsepchar_a{\loi_ifx{[\loi_nxttok}\setsepchar_b{\setsepchar_b[/]}}
\long\def\setsepchar_b[#1]#2{% #1=sepcar de <liste des sepcar>  #2=<liste des sepcar>
  \loi_ifempty{#1}
    {\loi_error{Empty separator not allowed, separator "/" used}%
    \setsepchar_b[/]{#2}%
    }
    {\def\loi_currentsep{#1}%
    \_removeextremespacesfalse
    \loi_nestcnt1 % réinitaliser niveau initial à 1
    \def\nestdepth{1}%
    \loi_argcsname\let{loi_previndex[\number\loi_nestcnt]}\empty
```

```
    \def\loi_listname{loi_listofsep}%
    \let\loi_def\def \let\loi_edef\edef \let\loi_let\let
    \let\loi_listofpair_saved\loi_list_ofpair
    \let\loi_list_ofpair\empty
    \loi_ifempty{#2}
      {\loi_error{Empty list of separators not allowed, "," used}%
      \readlist_e1{,}%
      }
      {\readlist_e1{#2}%
      }%
    \loi_argcsname\let\nestdepth{loi_listofseplen[0]}%
    \loi_argcsname\let\loi_currentsep{loi_listofsep[1]}% 1er car de séparation
    \let\loi_listofpair\loi_listofpair_saved
    }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% macro normalisant l'index %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\loi_normalizeindex#1#2#3{% #1=macroname  #2=liste d'index  #3=profondeur max --> renvoie {err↙
     }{indx norm}
  \loi_ifempty{#2}
    {\loi_stop{}{}}
    {\loi_normalizeindex_a1{}{#3}{#1}#2,\loi_quark,}%
}%
\def\loi_normalizeindex_a#1#2#3#4#5,{% #1=compteur de profondeur #2=index précédents  #3=profondeur↙
      max #4=macroname  #5=index courant
  \loi_ifx{\loi_quark#5}
    {\loi_normalizeindex_c#2\loi_quark% supprimer la dernière virgule
    }
    {\loi_ifnum{#1>#3 }
      {\loi_invalidindex{Too deeply nested index, index [.] retained}{#2}% si profondeur trop ↙
           grande
      }
      {\loi_ifinrange\ifnum\numexpr#5<0 -1*\fi(#5)[[1:\csname #4len[#20]\endcsname]]% si abs(#5) ↙
           hors de [1,len]
        {\loi_exparg\loi_normalizeindex_b{\number\numexpr#5\ifnum\numexpr#5<0 +\csname #4len[#20]\↙
             endcsname+1\fi}{#1}{#2}{#3}{#4}}
        {\loi_invalidindex{#5 is an invalid index, index [.] retained}{#2}}%
      }%
    }%
}
\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
\def\loi_normalizeindex_c#1,\loi_quark{\loi_stop{}{#1}}
\def\loi_invalidindex#1#2{\loi_ifempty{#2}{\loi_invalidindex_a{#1},}\loi_invalidindex_a{#1}{#2}}
\def\loi_invalidindex_a#1#2{\loi_invalidindex_b#1\loi_quark#2\loi_quark}
\def\loi_invalidindex_b#1[.]#2\loi_quark#3,\loi_quark#4\loi_quark,{\loi_stop{#1[#3]#2}{#3}}% #4= ↙
     index ignorés

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% macro publique \readlist %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcount\loi_nestcnt
\def\greadlist{\let\loi_def\gdef\let\loi_edef\xdef\def\loi_let{\global\let}\readlist_a}%
\def\readlist{\let\loi_def\def\let\loi_edef\edef\let\loi_let\let\readlist_a}
```

```
\def\readlist_a{%
  \loi_nestcnt1 % niveau initial = 1
  \loi_argcsname\let{loi_previndex[\number\loi_nestcnt]}\empty
  \loi_ifstar{\_removeextremespacestrue\readlist_b}{\_removeextremespacesfalse\readlist_b}%
}
\long\def\readlist_b#1#2{% #1=macro stockant les éléments  #2=liste des éléments
  \loi_ifcsexpandable{#2}
    {\loi_exparg{\readlist_b#1}{#2}%
    }
    {\loi_edef\loi_listname{\loi_macroname#1}%
    \loi_argcsname\loi_let{\loi_listname nest}\nestdepth
    \loi_argcsname\loi_def{\loi_listname[]}{#2}% la liste entière
    \loi_argcsname\loi_def{\loi_listname sep[]}{}% séparateur vide
    \loi_ifempty{#2}
      {\loi_def#1[##1]{}%
      \loi_argcsname\loi_def{\loi_listname len}{0}\loi_argcsname\loi_def{\loi_listname len[0]}{0}%
      \loi_error{Empty list ignored, nothing to do}%
      }
      {\loi_edef#1[##1]{\unexpanded{\romannumeral\expandafter\loi_checkindex\romannumeral\↙
           loi_normalizeindex}{\loi_listname}{##1}{\csname\loi_listname nest\endcsname}{\↙
           loi_listname}}%
      \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}}%
      \readlist_c{#2}%
      \loi_argcsname\loi_argcsname\loi_let{\loi_listname len}{\loi_listname len[0]}% longueur du ↙
           niveau 0
      }%
    }%
}
\def\loi_checkindex#1#2#3{%
  \expandafter\expandafter\expandafter\loi_stop\csname#3[#2]\expandafter\endcsname
  \romannumeral\loi_ifempty{#1}{\loi_stop}{\loi_stop\loi_error{#1}}%
}
\def\readlist_c{%
  \loi_argcsname\loi_let\loi_currentsep{loi_listofsep[\number\loi_nestcnt]}%
  \expandafter\readlist_d\loi_currentsep||\_nil
}
\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 dessous
\long\def\readlist_e#1#2{% #1=compteur d'index  #2=liste d'éléments à examiner terminée par <sep ↙
    courant simple> >>RIEN laissé après
  \loi_ifempty{#2}
    {\loi_argcsname\loi_edef{\loi_listname len[\csname loi_previndex[\number\loi_nestcnt]\endcsname↙
         0]}{\number\numexpr#1-1\relax}%
    \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
    \advance\loi_nestcnt-1
    \loi_argcsname\loi_let\loi_currentsep{loi_listofsep[\number\loi_nestcnt]}%
    }
    {\loi_expafter{\readlist_f{#2}{}}\loi_currentsep||\loi_quark||#2\_nil{#1}% aller isoler le 1er ↙
         item
    }%
}
\long\def\readlist_f#1#2#3||{% #1=liste restante   #2=<dernier sep utilisé>  #3=<sep courant>
  \loi_ifx{\loi_quark#3}% on a épuisé tous les <séparateurs> ? RESTE à lire <expr+sep1>\_nil{<↙
```

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

```
      \readlist_c{#3}% recommencer avec l'élément courant
      }
      {}%
    \loi_exparg\readlist_e{\number\numexpr#1+1}{#2}% puis chercher l'élément suivant dans la liste restante
    }%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% macro \listlen %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\listlen#1[#2]{%
  \romannumeral\loi_ifempty{#2}
    {\expandafter\expandafter\expandafter\loi_stop\csname\loi_macroname#1len[0]\endcsname}
    {\loi_exparg\listlen_a{\romannumeral-`\.\loi_macroname#1}{#2}}%
}
\def\listlen_a#1#2{% #1=macro name   #2=index non normalisé    prendre <profondeur max-1>
  \loi_exparg{\expandafter\listlen_b\romannumeral\loi_normalizeindex{#1}{#2}}{\number\numexpr\csname#1nest\endcsname-1}{#1}%
}
\def\listlen_b#1#2#3{% #1=err   #2=index normalisé  #3=macroname
  \expandafter\expandafter\expandafter\loi_stop\csname#3len[#2,0]\expandafter\endcsname
  \romannumeral\loi_ifempty{#1}{\loi_stop}{\loi_stop\loi_error{#1}}%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% macro \foreachitem %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\foreachitem#1\in#2{%
  \edef\foreachitem_a{\noexpand\foreachitem_c\noexpand#1{\expandafter\noexpand\csname\loi_macroname#1cnt\endcsname}{\loi_macroname#2}}%
  \futurelet\loi_nxttok\foreachitem_b
}
\def\foreachitem_b{\loi_ifx{\loi_nxttok[}\foreachitem_a{\foreachitem_a[]}}
\def\foreachitem_c#1#2#3[#4]{% prendre <profondeur max-1>
  \loi_exparg{\expandafter\foreachitem_d\romannumeral\loi_normalizeindex{#3}{#4}}{\number\numexpr\csname#3nest\endcsname-1}#1{#2}{#3}%
}
\def\foreachitem_d#1#2{\loi_ifempty{#2}{\foreachitem_e{#1}{}}{\foreachitem_e{#1}{#2,}}% #1=err   #2=index norm
\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
  \loi_ifnum{\csname#5len[#20]\endcsname>0 }
    {\loi_ifempty{#1}{}{\loi_error{#1}}%
    \loi_fornum#4=1to\csname#5len[#20]\endcsname\do{\loi_argcsname\let#3{#5[#2#4]}#6}%
    }
    {}%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% macro \showitem %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\showitems{\loi_ifstar{\let\showitems_cmd\detokenize\showitems_a}{\let\showitems_cmd\loi_identity\showitems_a}}
\def\showitems_a#1{\def\showitems_b{\showitems_d#1}\futurelet\loi_nxttok\showitems_c}
\def\showitems_c{\loi_ifx{\loi_nxttok[}\showitems_b{\showitems_b[]}}
```

```
\def\showitems_d#1[#2]{\foreachitem\showitems_ater\in#1[#2]{\showitemsmacro{\expandafter\↙
    showitems_cmd\expandafter{\showitems_ater}}}}
\unless\ifdefined\fbox
  \newdimen\fboxrule \newdimen\fboxsep \fboxrule=.4pt \fboxsep=3pt % réglages identiques à LaTeX
  \def\fbox#1{% imitation de la macro \fbox de LaTeX, voir pages 271 à 274 de "Apprendre à ↙
      programmer en TeX"
    \hbox{%
      \vrule width\fboxrule
      \vtop{%
        \vbox{\hrule height\fboxrule \kern\fboxsep \hbox{\kern\fboxsep#1\kern\fboxsep}}%
        \kern\fboxsep \hrule height\fboxrule
      }\vrule width\fboxrule
    }%
  }
\fi
\def\showitemsmacro#1{% encadrement par défaut
  \begingroup\fboxsep=0.25pt \fboxrule=0.5pt \fbox{\strut#1}\endgroup
  \hskip0.25em\relax
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% macro \itemtomacro %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\itemtomacro#1[#2]{% #1[#2]=item  non encore lu: #3=macro
  \edef\loi_listname{\loi_macroname#1}%
  \loi_exparg{\expandafter\itemtomacro_a\romannumeral\expandafter\loi_normalizeindex\expandafter{\↙
      loi_listname}{#2}}{\csname\loi_listname nest\endcsname}\let
}
\def\gitemtomacro#1[#2]{% #1[#2]=item
  \xdef\loi_listname{\loi_macroname#1}%
  \loi_exparg{\expandafter\itemtomacro_a\romannumeral\expandafter\loi_normalizeindex\expandafter{\↙
      loi_listname}{#2}}{\csname\loi_listname nest\endcsname}{\global\let}%
}
\def\itemtomacro_a#1#2#3#4{%
  \loi_ifempty{#1}{}{\loi_error{#1}}%
  \loi_argcsname#3#4{\loi_listname[#2]}%
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% réglages par défaut %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newif\if_removeextremespaces
\newif\if_ignoreemptyitems
\let\ignoreemptyitems\_ignoreemptyitemstrue
\let\reademptyitems\_ignoreemptyitemsfalse
\setsepchar{,}
\defpair{}
\loi_def_foreachsep{,}
\reademptyitems

\loi_restorecatcode
\endinput

#########################################################################
##########################  Historique ##################################
#########################################################################
```

```
v1.0  19/8/2016
  - Première version publique

v1.1  01/09/2016
  - Stockage des séparateurs dans <macrolist>sep
  - bug corrigé dans \loi_restorecatcode

v1.2  22/10/2016
  - macros \greadlist et \gitemtomacro pour la globalité

v1.3  18/11/2016
  - bugs corrigés dans la gestion de la globalité

v1.4  05/10/2017
  - test \loi_ifprimitive ajouté au test \loi_ifcs
  - suppression de \loi_expafternil, création de \loi_expafter,
    modification de \loi_argcsname
  - correction d'un bug : \setsepchar{\par} ne provoque plus
    d'erreur. \loi_ifnum devient \long

v1.5  06/10/2017
  - correction d'un bug dans \loi_ifcs

v1.51 24/10/2017
  - correction d'un bug dans \loi_ifcs

v1.52 13/01/2018
  - le dernier séparateur est <vide>

v1.53 13/03/2018
  - correction d'un bug dans \readlist_g

v1.6 01/11/2018
  - possibilité d'appariement de tokens dans les items

v1.61 03/03/2019
    - la macro \loi_ifcs contient une erreur de conception.
      Il faut tester si le token est un sc && s'il est
      développable pour renvoyer vrai car il existe des sc
      non développables && qui ne sont _pas_ des primitives.
      Macro rebaptisée \loi_ifcsexpandable
```