

L'extension

listofitems

v1.62
18 mai 2019

Christian TELLECHEA*
Steven B. SEGLETES†

Cette petite extension pour est destinée à lire une liste d'éléments dont le séparateur peut être choisi par l'utilisateur. Une fois la liste lue, ses éléments sont stockés dans une structure qui se comporte comme un tableau unidimensionnel et ainsi, il devient très facile d'accéder à un élément de la liste par son numéro. Par exemple, si la liste est stockée dans la macro `\foo`, l'élément n° 3 est désigné par `\foo[3]`. Un élément peut à son tour être une liste disposant d'un autre séparateur que celui de la liste de niveau supérieur, ce qui ouvre la voie à des imbrications et donne une syntaxe rappelant celle des tableaux à plusieurs dimensions du type `\foo[3,2]` pour accéder à l'élément n° 2 de la liste contenue dans l'élément n° 3 de la liste de plus haut niveau.

*. unbonpetit@netc.fr

†. steven.b.segletes.civ@mail.mil

1 Avant-propos

Important : à partir de la version 1.62, listofitems nécessite un moteur \TeX fournissant la primitive `\expanded`. Si ce n'est pas le cas, un message d'erreur sera émis et la version 1.61 sera chargée (dernière version fonctionnant sans la primitive `\expanded`) : il est vivement conseillé de mettre à jour sa distribution \LaTeX afin de profiter d'un moteur \TeX récent permettant l'utilisation de cette nouvelle primitive.

Cette extension ne requiert aucun package, doit être utilisée avec un moteur $\varepsilon\text{-}\TeX$, et doit être appelée sous `(pdf)(Xe)(lua) \LaTeX` par

```
\usepackage{listofitems}
```

et sous `(pdf)(Xe)(lua) \TeX` par

```
\input listofitems.tex
```

2 Lire une liste simple

Définir le séparateur Le *séparateur* par défaut est la virgule et si l'on souhaite en changer, il faut, avant de lire une liste d'éléments, le définir avec `\setsepchar{<séparateur>}`. Un *séparateur* est un ensemble de tokens dont les catcodes sont différents de 1 et 2 (les accolades ouvrantes et fermantes), 14 (habituellement %) et 15. Le token de catcode 6 (habituellement #) n'est accepté que s'il est suivi d'un entier auquel cas l'ensemble désigne l'argument d'une macro ; en aucun cas ce token ne doit se trouver seul dans le *séparateur*. Des commandes peuvent se trouver dans cet ensemble de tokens, y compris la primitive `\par`.

Le *séparateur* « / » est réservé par défaut pour les listes imbriquées (voir page 3). Il ne faut donc pas écrire « `\setsepchar{/}` » car listofitems comprendrait que l'on souhaite lire une liste imbriquée. Pour définir « / » comme *séparateur* d'une liste simple, il faut, à l'aide de l'argument optionnel, choisir un autre *séparateur* de listes imbriquées, par exemple « . » et écrire « `\setsepchar[.]{/}` ».

Il n'est pas possible de choisir « | » comme *séparateur* car il entrerait en conflit avec l'opérateur logique **OU** noté « || » (voir plus bas). On peut cependant contourner cette limitation, à ses risques et périls, en écrivant « `\setsepchar{|}` ».

Lire la liste Pour lire la liste d'éléments, la commande `\readlist<macroliste>{<liste>}` doit être appelée. Ce faisant, la *liste* est lue et les éléments sont stockés dans une macro, notée *macroliste* qui dès lors, se comporte comme un tableau composé des éléments de la *liste*. Un élément est un ensemble de tokens dont les accolades *doivent* être équilibrées. Les tokens de catcode 6 seuls, 14 et 15 ne sont pas autorisés dans les listes.

Par exemple, pour définir la *macroliste* nommée `\foo`, on peut écrire

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

Si la *liste* est contenue dans une macro, alors cette macro est développée par listofitems. On peut donc écrire `\readlist<macroliste>{<macro>}` ce qui donnerait

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

La macro `\greadlist` agit comme `\readlist` mais effectue des assignations *globales* et par conséquent, la *macroliste* est utilisable hors du groupe où a été exécutée `\greadlist`.

Accéder à un élément La macro `\foo` attend un argument numérique *obligatoire* entre crochets, que nous notons *i* et qui désigne le rang de l'élément auquel on souhaite accéder. Ainsi, `\foo[1]` est³ « 12 ». De la même façon, `\foo[4]` est « `{\bfseries z}` ».

Le nombre *i* peut également être négatif auquel cas le comptage se fait à partir de la fin de la liste : -1 représente le dernier rang, -2 l'avant-dernier, etc. Si le nombre d'éléments est *n*, alors l'argument `-n` est le premier élément.

3. Il faut 2 développements à `\foo[i]` pour obtenir l'élément n° *i*.

D'une façon générale, si une $\langle liste \rangle$ a une longueur n , alors l'index i peut se trouver dans l'intervalle $[1; n]$ ou $[-n; -1]$ et dans le cas contraire, une erreur de compilation survient. Si l'index est vide, alors $\backslash foo[]$ se développe en la $\langle liste \rangle$ entière.

Accéder à un séparateur Lorsque $\backslash readlist\foo\langle liste \rangle$ est exécuté, la macro $\backslash foosep$ est créée. Elle s'utilise avec la syntaxe $\backslash foosep[\langle index \rangle]$ et permet d'accéder au séparateur qui suit l'élément de rang $\langle index \rangle$. Le dernier séparateur (celui qui suit le dernier élément) est vide. Si l' $\langle index \rangle$ est vide, $\backslash foosep[]$ a un développement vide.

Choisir plusieurs séparateurs possibles Pour spécifier plusieurs séparateurs possibles, il faut utiliser l'opérateur **OU** noté « $||$ ». On peut par exemple utiliser cette fonctionnalité pour isoler les termes dans une somme algébrique :

$\backslash setsepchar\{+ - \}$	
$\backslash readlist\terme\{17-8+4-11\}$	1) 17 (séparateur = -)
1) $\backslash terme[1]$ (séparateur = $\backslash termesep[1]$) $\backslash par$	2) 8 (séparateur = +)
2) $\backslash terme[2]$ (séparateur = $\backslash termesep[2]$) $\backslash par$	3) 4 (séparateur = -)
3) $\backslash terme[3]$ (séparateur = $\backslash termesep[3]$) $\backslash par$	4) 11 (séparateur =)
4) $\backslash terme[4]$ (séparateur = $\backslash termesep[4]$)	

Nombre d'éléments Si l'on écrit $\backslash readlist\langle macroliste \rangle\langle liste \rangle$ alors la macro $\langle macroliste \rangle len$ contient⁴ le nombre d'éléments de la $\langle liste \rangle$. Dans l'exemple avec $\backslash foo$, la macro $\backslash foolen$ contient 8.

Afficher tous les éléments À des fins de débogage, la macro $\backslash showitems\langle macroliste \rangle$ compose tous les éléments d'une liste tandis que sa version étoilée affiche ces éléments « détokénisés⁵ ».

$\backslash showitems\foo\par$	$\boxed{12} \boxed{abc} \boxed{x y} \boxed{z} \boxed{\TeX} \boxed{!}$
$\backslash showitems*\foo$	$\boxed{12} \boxed{abc} \boxed{x y} \boxed{\{\bfseries z\}} \boxed{\TeX} \boxed{!}$

La présentation de chaque élément est confiée à la macro $\backslash showitemsmacro$ dont le code est

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

Il est donc possible — et souhaitable — de la redéfinir si l'on cherche un autre effet.

La macro $\backslash fbox$ et ses dimensions afférentes $\backslash fboxsep$ et $\backslash fboxrule$ sont définies par $\backslash listofitems$ lorsqu'on ne compile pas sous \LaTeX de façon à obtenir le même résultat qu'avec \TeX .

Suppression des espaces extrêmes Par défaut, $\backslash listofitems$ lit et prend en compte le (ou les) espaces se trouvant au début et à la fin d'un élément. Pour que ces espaces soient ignorés lors de la lecture de la $\langle liste \rangle$, il faut exécuter la version étoilée $\backslash readlist*\langle macro \rangle\langle liste \rangle$:

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

Gestion des éléments vides Par défaut, $\backslash listofitems$ prend en compte les éléments vides. Ainsi, dans l'exemple précédent, le 2-développement de $\backslash foo[7]$ est vide. Pour que des éléments vides (ceux délimités par deux séparateurs consécutifs dans la liste) soient ignorés, il faut, avant de lancer la macro $\backslash readlist$, exécuter la macro $\backslash ignoreemptyitems$. La macro $\backslash reademptyitems$ revient au comportement par défaut.

4. C'est-à-dire qu'elle est purement développable et se développe en un nombre
5. La primitive $\backslash detokenize$ qui procède à cette dénaturation insère un espace après chaque séquence de contrôle.

Cette option peut être utilisée seule ou combinée avec `\readlist*` auquel cas la suppression des espaces extrêmes intervient *avant* que `listofitems` n'ignore les éléments vides :

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

Itérer sur la liste Une fois une liste lue par `\readlist` et stockée dans une *macro* liste, la commande `\foreachitem` *variable* `\in` *macro* liste `{code}` itère sur la liste : la *variable* est une macro choisie par l'utilisateur qui prendra tour à tour la valeur de chaque élément. La macro *variable*cnt représente le numéro de l'élément contenu dans *variable*.

<pre>\setsepchar{ }% séparateur = espace \readlist\phrase{Une phrase de test.} \foreachitem\mot\in\phrase{Le mot numéro \motcnt{} : \mot\par}</pre>	<pre>Le mot numéro 1 : Une Le mot numéro 2 : phrase Le mot numéro 3 : de Le mot numéro 4 : test.</pre>
---	--

Assigner un élément à une macro La commande `\itemtomacro` *macro* liste `[index]` *macro* assigne à la *macro* l'élément désigné par *macro* liste `[index]`. La *macro* ainsi définie est purement développable, sous réserve que l'élément qu'elle contient le soit.

<pre>\setsepchar{ }% séparateur = espace \readlist\phrase{Une phrase de test.} \itemtomacro\phrase[2]\unmot \meaning\unmot\par \itemtomacro\phrase[-1]\motdelafin \meaning\motdelafin</pre>	<pre>macro:->phrase macro:->test.</pre>
---	---

3 Listes imbriquées

On parle de liste « imbriquée » lorsque l'on demande à `listofitems` de lire une liste où les éléments sont à leur tour compris comme une liste (dont le séparateur est différent de la liste de niveau supérieur). Le nombre d'imbrication n'est pas limité, mais dans la pratique, un niveau d'imbrication de 2, voire 3, semble un maximum.

Définir les séparateurs Pour indiquer que les éléments de la liste doivent eux-mêmes être compris comme des listes et que la recherche des éléments sera récursive, il faut spécifier plusieurs *séparateurs*, chacun correspondant à un niveau d'imbrication. Pour déclarer une *liste de séparateurs* il faut définir le *séparateur* de cette *liste de séparateurs* à l'aide de l'argument optionnel de la macro `\setsepchar` et écrire `\setsepchar[separateur]{liste des séparateurs}`. Par défaut, le *séparateur* est `</>`. Ainsi, si l'on donne l'ordre

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

on indique une profondeur récursive de 3 et on choisit comme séparateur de la *liste des séparateurs* le caractère par défaut `</>` :

- les éléments de niveau 1 sont trouvés entre les séparateurs `<\/>` ;
- les éléments de niveau 2 sont trouvés dans les éléments de niveau 1 entre les séparateurs `< , >` ;
- enfin, les éléments de niveau 3 sont trouvés dans ceux de niveau 2 entre les séparateurs `< _ >`.

La *profondeur* de recherche est contenue dans la macro purement développable `\nestdepth`.

Lire et accéder aux éléments Pour les listes imbriquées, les index obéissent à la règle suivante :

- [] désigne la liste principale, c'est-à-dire l'argument de `\readlist` ;
- [*i*] désigne l'élément n° *i* de la liste principale ;
- [*i*], [*j*] désigne l'élément n° *j* de la liste constituée par l'élément évoqué au point précédent ;
- [*i*], [*j*], [*k*] désigne l'élément n° *k* de la liste constituée par l'élément évoqué au point précédent ;
- etc.

Comme pour les liste non imbriquées, les index peuvent être négatifs.

Pour lire les éléments, la syntaxe de `\readlist` est exactement la même qu'avec les listes simples :

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

L'opérateur « || » Cet opérateur peut se trouver dans n'importe quel niveau d'imbrication.

<code>\setsepchar[,]{+ -,* /}</code>	
<code>\readlist\nombres{1+2*3-4/5*6}</code>	
Terme 1 : <code>\nombres[1]</code> \par	Terme 1 : 1
Terme 2 : <code>\nombres[2]</code> (facteurs : <code>\nombres[2,1]</code> et <code>\nombres[2,2]</code>)\par	Terme 2 : 2*3 (facteurs : 2 et 3)
Terme 3 : <code>\nombres[3]</code> (facteurs : <code>\nombres[3,1]</code> , <code>\nombres[3,2]</code> et <code>\nombres[3,3]</code>)	Terme 3 : 4/5*6 (facteurs : 4, 5 et 6)

Nombre d'éléments La macro `\listlen<macrolist>[<index>]` nécessite 2 développements pour donner le nombre d'éléments de la liste spécifiée par l'<index>.

La <profondeur> de l'<index> doit être strictement inférieure à celle de la <liste>.

Dans le cas d'un <index> vide, `\listlen<macrolist>[]` donne en 2 développements le même résultat que `<macrolist>len` qui le donne en 1.

<code>\setsepchar{\,/ }</code>	
<code>\readlist\baz{1,2 a b,3 c\4 d e f,5,6\7,,8, ,9 xy z}</code>	a) 3 ou 3
a) <code>\bazlen\</code> ou <code>\listlen\baz[]</code> \par	b) 3
b) <code>\listlen\baz[1]</code> \par	c) 3
c) <code>\listlen\baz[2]</code> \par	d) 5
d) <code>\listlen\baz[3]</code> \par	e) 1
e) <code>\listlen\baz[3,1]</code> \par	f) 2
f) <code>\listlen\baz[3,4]</code> \par% 2 éléments vides	g) 3
g) <code>\listlen\baz[3,5]</code>	

Afficher les éléments La macro `\showitems<macrolist>[<index>]` affiche les éléments de la liste spécifiée par <index>, selon le même principe que `\listlen`.

La <profondeur> de l'<index> doit être strictement inférieure à celle de la <liste>.

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

Éléments vides et espaces extrêmes La suppression des éléments vides et/ou des espaces extrêmes intervient dans *tous* les éléments, quel que soit le degré d'imbrication. Il est clair que choisir un espace comme séparateur est inutile si l'on veut utiliser `\readlist*`. C'est pourquoi dans cet exemple, « * » est choisi comme séparateur.

Dans cet exemple, on ne supprime que les espaces extrêmes en gardant les éléments vides.

<code>\setsepchar{\,/,/*}</code>	a) $\overline{1, 2^*a^*b .3^*c} \overline{4^*d^*e^*f,5,6} \overline{7,,8, .9^* xy ^*z}$
<code>\readlist*\baz{1, 2*a*b ,3*c\4*d*e*f,5,6\7,,8, ,9* xy *z}</code>	b) $\overline{1} \overline{2^*a^*b} \overline{3^*c}$
a) <code>\showitems\baz[]\par</code>	c) $\overline{4^*d^*e^*f} \overline{5} \overline{6}$
b) <code>\showitems\baz[1]\par</code>	d) $\overline{7} \overline{8} \overline{9^* xy ^*z}$
c) <code>\showitems\baz[2]\par</code>	e) $\overline{7}$
d) <code>\showitems\baz[3]\par</code>	f) $\overline{\quad}$
e) <code>\showitems\baz[3,1]\par</code>	g) $\overline{9} \overline{xy} \overline{z}$
f) <code>\showitems\baz[3,4]\par</code>	
g) <code>\showitems\baz[3,5]% "xy" sans espaces extrêmes</code>	

Itérer sur une liste La syntaxe `\foreachitem <variable> \in <macro>[<index>]{<code>}` reste valable où désormais, l'<index> spécifie sur quel élément (compris comme une liste) on veut itérer.

La <profondeur> de l'<index> doit être strictement inférieure à celle de la <liste>.

Assigner un élément à une macro La syntaxe `\itemtomacro<macroliste>[<index>]<macro>` reste valable pour assigner à <macro> l'élément spécifié par <macroliste>[<index>].

<code>\setsepchar[,]{\, , }</code>	
<code>\readlist\poeme{J'arrive tout couvert encore de rosée\\% Que le vent du matin vient glacer à mon front.\\% Souffrez que ma fatigue à vos pieds reposée\\% Rêve des chers instants qui la délasseront.}% 2e strophe de « Green », Paul Verlaine</code>	
<code>\itemtomacro\poeme[2]\vers</code>	2e vers = Que le vent du matin vient glacer à mon front.
<code>2e vers = \vers</code>	Un mot = glacer
<code>\itemtomacro\poeme[2,-4]\mot</code>	
<code>Un mot = \mot</code>	

La macro `\gitemtomacro` fait une assignation globale.

4 Tokens appariés

Pour le découpage des items, il est possible à partir de la version 1.6, de tenir compte de la présence de caractères *appariés*. Ainsi, si une liste de caractères appariés est définie, chaque item s'étend jusqu'au prochain (<separateur>) qui équilibre les tokens appariés.

Pour définir une liste de tokens appariés, on utilise

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

où les tokens sont lus deux par deux pour former les paires d'appariement. Un <token> d'appariement doit être constitué d'un seul caractère; les macros, primitives, espaces, accolades, token «#», ensembles de plusieurs tokens entre accolades sont interdits. Deux tokens formant une paire *doivent* être différents.

<code>\setsepchar{+ -}</code>	
<code>\defpair{()[]}</code>	
<code>\readlist\termes{1+2*[3+4*(5+6-7)+8]-9+10}</code>	$\overline{1} \overline{2^*[3+4*(5+6-7)+8]} \overline{9} \overline{10}$
<code>\showitems\termes</code>	

Pour revenir au comportement par défaut, c'est-à-dire sans tokens appariés, il faut exécuter

```
\defpair{}
```

Dans une expression, pour stocker dans une macro ce qui se trouve entre deux tokens appariés, on peut faire appel à

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

qui mettra dans la `\macro` ce qui se trouve entre la paire $\langle tok1 \rangle \langle tok2 \rangle$ dans l' $\langle expression \rangle$.

```

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

\itemtomacro\termes[2]\parenterm
Dans la parenthèse extérieure :
\insidepair()\parenterm\inbigparen
"\inbigparen"

Dans la parenthèse intérieure :
\insidepair()\inbigparen\inmallparen
"\inmallparen"

```

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

Dans la parenthèse extérieure : "3+4*(5+6-7)+8"
 Dans la parenthèse intérieure : "5+6-7"

5 Le code

Toute suggestion, remontée de bug, remarque, demande, ajout ou modification de fonctionnalité est bienvenue ; dans ce cas, j'invite les utilisateurs de `listofitems` à m'envoyer un email à `unbonpetit@netc.fr`.

Le code ci-dessous est l'exact verbatim du fichier `listofitems.tex`. J'espère que les quelques commentaires qui le parsèment de-ci de-là seront suffisants pour que l'utilisateur ou le curieux comprenne la machinerie interne de ce 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            {1.62}
8 %
9 \def\loidate           {2019/05/18}
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 2016-2019
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
33 \expandafter\edef\csname loi_restorecatcode\endcsname{\catcode\number'_{\_}=\number\catcode'_{\_}
34 \relax}
35 \catcode'_{\_}11
36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

37 %%% gestion des erreurs + annonce package %%%
38 %%%
39 \unless\ifdefined\loi_fromsty
40 \immediate\write -1 {Package: \loidate\space v\loiver\space Grab items in lists using user-
specified sep char (CT)}%
41 \fi
42
43 \ifdefined\PackageError
44 \def\loi_error#1{\PackageError\loiname{#1}{Read the \loiname\space 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 des prérequis %%%
51 %%%
52 \def\loi_checkprimitive#1#2#3{% Vérifie que #1 est une primitive et sinon, émet le message #2
et exécute #3
53 \begingroup
54 \edef__tempa{meaning#1}\edef__tempb{string#1}\expandafter
55 \endgroup
56 \ifx__tempa__tempb\else
57 \loi_error{#2}%
58 \def\loi_temp{#3}%
59 \loi_restorecatcode\expandafter\loi_temp
60 \fi
61 }
62 \loi_checkprimitive\eTeXversion
63 {You are not using an eTeX engine, listofitems cannot work.}
64 {\endinput}%
65 \loi_checkprimitive\expanded
66 {the \string\expanded\space primitive is not provided by your TeX engine, listofitems v\
loiver\space cannot work: loading listofitems v1.61}
67 {\input listofitemsold.tex\relax\endinput}%
68
69 %%%
70 %%% macros auxiliaires %%%
71 %%%
72 \def\loi_quark{\loi_quark}
73 \long\def\loi_identity#1{#1}
74 \long\def\loi_gobarg#1{}
75 \long\def\loi_first#1#2{#1}
76 \long\def\loi_second#1#2{#2}
77 \long\def\loi_firsttonil#1#2\__nil{#1}
78 \long\def\loi_antefi#1#2\fi{#2\fi#1}
79 \long\def\loi_exparg#1#2{\expandafter\loi_exparg_a\expandafter{#2}{#1}}% \loi_exparg{<a>{<b>}
devient <a>{<*>}
80 \long\def\loi_exparg_a#1#2{#2{#1}}
81 \long\def\loi_expafter#1#2{\expandafter\loi_expafter_a\expandafter{#2}{#1}}% \loi_expafter{<a
>}{<b>} devient <a><*>
82 \long\def\loi_expafter_a#1#2{#2#1}
83 \def\loi_macroname{\loi_ifinrange\escapechar[[0:255]]{\expandafter\loi_gobarg}{}\string}
84 \def\loi_argcsname#1{\loi_argcsname_a{#1}}
85 \def\loi_argcsname_a#1#2{\loi_expafter{#1}{\csname#2\endcsname}}
86 \long\def\loi_addtomacro#1#2{\loi_exparg{\def#1}{#1#2}}
87
88 %%%
89 %%% macros de test %%%
90 %%%
91 \long\def\loi_ifnum#1{\ifnum#1\expandafter\loi_first\else\expandafter\loi_second\fi}
92 \long\def\loi_ifx#1{\ifx#1\expandafter\loi_first\else\expandafter\loi_second\fi}

```



```

93 \long\def\loi_ifempty#1{\loi_exparg\loi_ifx{\expandafter\relax\detokenize{#1}\relax}}
94 \def\loi_ifstar#1#2{\def\loi_ifstar_a{\loi_ifx{*}\loi_nxttok}{\loi_first{#1}}{#2}}\futurelet\
loi_nxttok\loi_ifstar_a}
95 \edef\loi_escapechar{\expandafter\loi_gobarg\string\}
96 \long\def\loi_ifcsexpandable#1{% #1 est-il constitué d'une seule sc_développable ?
97   \loi_ifempty{#1}
98   {\loi_second
99   }
100   {\loi_ifspacefirst{#1}
101   {\loi_second% si espace en 1er, faux
102   }
103   {\csname loi_\if\loi_escapechar\expandafter\loi_firsttonil\detokenize{#1}\_nil first\else
second\fi\endcsname
104   {\loi_exparg\loi_ifempty{\loi_gobarg#1}% 1 seul arg commençant par "\" ?
105   {\def\loi_tempa{#1}\loi_exparg{\def\loi_tempb}{#1}% est-il développable ?
106   \expandafter\unless\loi_ifx{\loi_tempa\loi_tempb}%
107   }
108   {\loi_second
109   }%
110   }
111   {\loi_second
112   }%
113   }%
114   }%
115 }
116 \def\loi_ifinrange#1[[#2:#3]]{\expandafter\unless\loi_ifnum{\numexpr(#1-#2)*(#1-#3)>0 }}
117 \def\loi_ifstring#1\in#2{% si la chaîne #1 est contenue dans #2
118   \def\loi_ifstring_a##1#1##2\_nil{\loi_ifempty{##2}\loi_second\loi_first}%
119   \loi_ifstring_a#2#1\@nil% appel de la macro auxiliaire
120 }
121
122 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
123 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macro \loi_foreach %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
124 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
125 \newcount\loi_cnt_foreach_nest \loi_cnt_foreach_nest=0
126 \def\end_foreach{\end_foreach}
127 \def\loi_def_foreachsep#1{%
128   \long\def\loi_foreach##1\in##2##3{%
129     \global\advance\loi_cnt_foreach_nest1
130     \loi_argcsname\def{loop_code_\number\loi_cnt_foreach_nest}{##3}%
131     \loi_foreach_a##1##2#1\end_foreach#1%
132     \loi_argcsname\let{loop_code_\number\loi_cnt_foreach_nest}\empty
133     \global\advance\loi_cnt_foreach_nest-1
134   }%
135   \long\def\loi_foreach_a##1##2#1{%
136     \def##1{##2}%
137     \loi_ifx{\end_foreach##1}
138     {}
139     {\csname loop_code_\number\loi_cnt_foreach_nest\endcsname% exécute le code
140     \loi_foreach_a##1%
141     }%
142   }%
143 }
144
145 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
146 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macros gérant l'appariement %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
147 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
148 \long\def\defpair#1{%
149   \let\loi_listofpair\empty
150   \loi_ifempty{#1}
151   {}

```

```

152     {\defpair_a}{#1\loi_quark\loi_quark}%
153 }
154 \long\def\defpair_a#1#2#3{%
155   \loi_ifx{\loi_quark#2}
156   {\def\loi_sanitizelist##1,\_nil{\def\loi_listofpair{##1}}%
157   \loi_sanitizelist#1\_nil
158   }
159   {\loi_if_validpair#2#3%
160     {\long\def\loi_paired_a{#2}\long\def\loi_paired_b{#3}%
161     \loi_ifx{\loi_paired_a\loi_paired_b}
162     {\loi_error{Paired tokens must not be equal, the pair \detokenize{#2#3} is ignored}%
163     \defpair_a{#1}%
164     }
165     {\defpair_a{#1#2#3,}%
166     }%
167   }
168   {\loi_error{Invalid paired tokens, the pair "\detokenize{#2}" and "\detokenize{#3}" is
169   ignored}%
170   \defpair_a{#1}%
171   }%
172 }
173 \long\def\loi_if_validpair#1#2{%
174   \def\loi_validpair{1}%
175   \loi_if_invalid_paiRToken{#1}{\def\loi_validpair{0}}%
176   \loi_if_invalid_paiRToken{#2}{\def\loi_validpair{0}}%
177   \loi_ifnum{\loi_validpair=1 }
178 }
179 \long\def\loi_if_invalid_paiRToken#1{%
180   \loi_ifempty{#1}
181   {\loi_identity
182   }
183   {\loi_ifspacefirst{#1}
184     {\loi_identity
185     }
186     {\loi_exparg\loi_ifempty{\loi_gobarg#1}% 1 seul token ?
187     {\ifcat\relax\noexpand#1\expandafter\loi_identity\else\expandafter\loi_gobarg\fi}
188     {\loi_identity}% si plusieurs tokens, faux
189     }%
190   }%
191 }
192 \long\def\loi_count_occur#1\in#2:#3{% compte le nombre d'occurrences de #1 dans #2 et met le
193   résultat dans la macro #3
194   \long\def\loi_count_occur_a##1##2#1##3\_nil{%
195     \loi_ifempty{##3}
196     {\def#3{##1}}
197     {\expandafter\loi_count_occur_a\number\numexpr##1+1\relax##3\_nil}%
198   }%
199   \loi_count_occur_a0#2#1\_nil
200 }
201 \long\def\loi_check_pair#1#2\in#3{% teste l'appariement de #1 et #2 dans #3
202   \loi_ifempty{#3}
203   {\loi_second
204   }
205   {\loi_count_occur#1\in#3:\loi_tempa
206   \loi_count_occur#2\in#3:\loi_tempb
207   \loi_ifnum{\loi_tempa=\loi_tempb\relax}%
208   }%
209 }
210 \long\def\loi_grabpaired_expr#1#2#3#4#5{% #1=liste de paires #2=expression #3=séparateur
211   #4=résultat #5=ce qui reste

```

```

210 \let#4\empty
211 \def\loi_remain{#2#3}%
212 \loi_foreach\loi_pair\in{#1}{\expandafter\loi_grabpaired_expr_a\loi_pair{#3}#4}%
213 \def\loi_remove_lastsep##1#3\_nil{\def#4{##1}}%
214 \expandafter\loi_remove_lastsep#4\_nil
215 \loi_expafter{\long\def\loi_grab_remain}#4##1\_nil{%
216   \loi_ifempty{##1}
217   {\let#5\empty}
218   {\loi_exparg{\def#5}{\loi_gobarg##1}}%
219 }%
220 \loi_grab_remain#2\_nil
221 }
222 \long\def\loi_grabpaired_expr_a#1#2#3#4{% #1#2=paire en cours #3=séparateur #4=résultat
223 \loi_exparg{\loi_check_pair#1#2\in}#4% si les paires sont appariées dans le résultat
224 {}% passer à la paire suivante
225 {\long\def\loi_grabpaired_expr_b##1#3##2\_nil{%
226   \loi_addtomacro#4{##1#3}% ajouter au résultat ce qui est jusqu'au prochain séparateur
227   \def\loi_remain{##2}%
228   \loi_exparg{\loi_check_pair#1#2\in}{#4}
229   {}
230   {\loi_ifempty{##2}
231     {\loi_error{"\detokenize{#1}" and "\detokenize{#2}" are not paired}}
232     {\loi_grabpaired_expr_b##2\_nil}}%
233   }%
234 }%
235 \expandafter\loi_grabpaired_expr_b\loi_remain\_nil
236 }%
237 }
238 \def\insidepair#1#2#3#4{% #1#2=paire #3=expr #4=macro recevant le resultat
239 \loi_if_validpair#1#2%
240 {\loi_ifcsexpandable{#3}
241  {\loi_exparg{\insidepair#1#2}{#3}#4%
242  }
243  {\loi_check_pair#1#2\in{#3}% si les paires sont appariées dans le résultat
244   {\def\insidepair_a##1#1##2\_nil{\insidepair_b##2\_nil{#1}}%
245   \def\insidepair_b##1#2##2\_nil##3#3{%
246     \loi_check_pair#1#2\in{##3##1#2}
247     {\loi_exparg{\def#4}{\loi_gobarg##3##1}}%
248     {\insidepair_b##2\_nil{##3##1#2}}%
249   }%
250   \insidepair_a#3\_nil
251   }
252   {\loi_error{"\detokenize{#1}" and "\detokenize{#2}" are not paired in "#3"}%
253   }%
254 }%
255 }
256 {\loi_error{Invalid paired tokens "\detokenize{#1}" and "\detokenize{#2}", empty \string#4 ↵
257   returned}% et bim
258 \let#4\empty% voilà, bien fait pour vos gueules
259 }%
260 }
261 %%%
262 %%% macro \loi_fornum %%%
263 %%%
264 \def\loi_fornum#1=#2to#3\do{%
265   \edef#1{\number\numexpr#2}%
266   \expandafter\loi_fornum_a
267   \cname loi_fornum_\string#1\expandafter\endcname\expandafter
268   {\number\numexpr#3\expandafter}%
269   \expanded{\ifnum#1<\numexpr#3\relax>+\else<-\fi}%

```

```

270 #1%
271 }
272 \long\def\loi_fornum_a#1#2#3#4#5#6{%
273   \def#1{%
274     \unless\ifnum#5#3#2\relax
275       \loi_antefi{#6\edef#5{\number\numexpr#5#41\relax}#1}%
276     \fi}%
277 #1%
278 }
279
280 %%%%%%%%%%
281 %%%%%%%%%% macro retirant les espaces extrêmes %%%%%%%%%%
282 %%%%%%%%%%
283 \long\def\loi_ifspacefirst#1{\expandafter\loi_ifspacefirst_a\detokenize{#10} \_nil}
284 \long\def\loi_ifspacefirst_a#1 #2\_nil{\loi_ifempty{#1}}
285 \loi_expafter{\def\loi_gobspace}\space{}
286 \def\loi_removefirstspaces#1{\loi_ifspacefirst{#1}\loi_exparg\loi_removefirstspaces{\
loi_gobspace#1}}{\unexpanded{#1}}
287 \begingroup
288 \catcode0 12
289 \long\gdef\loi_removefirstspaces#1{\loi_removefirstspaces_a#1^^00 ^^00\_nil}
290 \long\gdef\loi_removefirstspaces_a#1 ^^00{\loi_removefirstspaces_b#1^^00}
291 \long\gdef\loi_removefirstspaces_b#1^^00#2\_nil{\loi_ifspacefirst{#2}\loi_removefirstspaces_a
#1^^00 ^^00\_nil}{\unexpanded{#1}}
292 \endgroup
293 \long\def\loi_removeextremespaces#1{\expanded{\loi_exparg\loi_removefirstspaces{\expanded{\
loi_removefirstspaces{#1}}}}}
294
295 %%%%%%%%%%
296 %%%%%%%%%% macro publique \setsepchar %%%%%%%%%%
297 %%%%%%%%%%
298 \def\setsepchar{\futurelet\loi_nxttok\setsepchar_a}
299 \def\setsepchar_a{\loi_ifx{[\loi_nxttok}\setsepchar_b{\setsepchar_b[/]}]}
300 \long\def\setsepchar_b[#1]#2{% #1=sepcar de <liste des sepcar> #2=<liste des sepcar>
301   \loi_ifempty{#1}
302     {\loi_error{Empty separator not allowed, separator "/" used}%
303     \setsepchar_b[/]{#2}%
304     }
305     {\def\loi_currentsep{#1}%
306     \_removeextremespacesfalse
307     \loi_nestcnt1 % réinitialiser niveau initial à 1
308     \def\nestdepth{1}%
309     \loi_argcsname\let\loi_previndex[\number\loi_nestcnt]\empty
310     \def\loi_listname{\loi_listofsep}%
311     \let\loi_def\def \let\loi_edef\edef \let\loi_let\let
312     \let\loi_listofpair_saved\loi_list_ofpair
313     \let\loi_list_ofpair\empty
314     \loi_ifempty{#2}
315       {\loi_error{Empty list of separators not allowed, "," used}%
316       \readlist_g1{,}%
317       }
318       {\readlist_g1{#2}%
319       }%
320     \loi_argcsname\let\nestdepth{\loi_listofseplen[0]}%
321     \loi_argcsname\let\loi_currentsep{\loi_listofsep[1]}% ler car de séparation
322     \let\loi_listofpair\loi_listofpair_saved
323   }%
324 }
325
326 %%%%%%%%%%
327 %%%%%%%%%% macro normalisant l'index %%%%%%%%%%

```

```

328 %%%%%%%%%%
329 \def\loi_normalizeindex#1#2{% #1=macroname #2=liste d'index --> renvoie {err}{indx norm}
330 \loi_ifempty{#2}
331 {}{}}
332 {\loi_exparg{\loi_normalizeindex_a1{}}{\number\csname#1nest\endcsname}{#1#2,\loi_quark,}%
333 }%
334 \def\loi_normalizeindex_a#1#2#3#4#5,{% #1=compteur de profondeur #2=index précédents #3=
profondeur max #4=macroname #5=index courant
335 \loi_ifx{\loi_quark#5}
336 {\loi_normalizeindex_c#2\loi_quark% supprimer la dernière virgule
337 }
338 {\loi_ifnum{#1>#3 }
339 {\loi_invalidindex{Too deeply nested index, index [.] retained}{#2}% si profondeur trop
grande
340 }
341 {\loi_ifinrange\ifnum\numexpr#5<0 -1*\fi(#5)[[1:\csname #4len[20]\endcsname]]% si abs
(#5) hors de [1,len]
342 {\loi_exparg\loi_normalizeindex_b{\number\numexpr#5\ifnum\numexpr#5<0 +\csname #4len[
20]\endcsname+1\fi}{#1}{#2}{#3}{#4}}
343 {\loi_invalidindex{#5 is an invalid index, index [.] retained}{#2}}%
344 }%
345 }%
346 }
347 \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
348 \def\loi_normalizeindex_c#1,\loi_quark{#1}{#1}}
349 \def\loi_invalidindex#1#2{\loi_ifempty{#2}{\loi_invalidindex_a{#1},}\loi_invalidindex_a{
#1}{#2}}
350 \def\loi_invalidindex_a#1#2{\loi_invalidindex_b#1\loi_quark#2\loi_quark}
351 \def\loi_invalidindex_b#1[.]#2\loi_quark#3,\loi_quark#4\loi_quark,{{#1[#3]#2}{#3}}% #4= index
ignorés
352
353 %%%%%%%%%%
354 %%%%%%%%%% macro publique \readlist %%%%%%%%%%
355 %%%%%%%%%%
356 \newcount\loi_nestcnt
357 \def\greadlist{\let\loi_def\gdef\let\loi_edef\xdef\def\loi_let{\global\let}\readlist_a}%
358 \def\readlist{\let\loi_def\def\let\loi_edef\edef\let\loi_let\let\readlist_a}
359 \def\readlist_a{%
360 \loi_nestcnt1% niveau initial = 1
361 \loi_argcsname\let{\loi_previndex[\number\loi_nestcnt]}\empty
362 \loi_ifstar{\_removeextremespacestrue\readlist_b}{\_removeextremespacesfalse\readlist_b}%
363 }
364 \long\def\readlist_b#1#2{% #1=macro stockant les éléments #2=liste des éléments
365 \loi_ifcsexpandable{#2}
366 {\loi_exparg{\readlist_b#1}{#2}%
367 }
368 {\loi_edef\loi_listname{\loi_macroname#1}%
369 \loi_exparg{\readlist_c#1{#2}}{\loi_listname}%%
370 }%
371 }
372 \long\def\readlist_c#1#2#3{% #1=macro stockant les éléments #2=liste des éléments #3=macroname
373 \loi_argcsname\loi_let{#3nest}\nestdepth
374 \loi_argcsname\loi_def{#3[]}{#2}% la liste entière
375 \loi_argcsname\loi_def{#3sep[]}{#2}% séparateur vide
376 \loi_ifempty{#2}
377 {\loi_def#1[#1]}%
378 \loi_argcsname\loi_def{#3len}{0}\loi_argcsname\loi_def{#3len[0]}{0}%
379 \loi_error{Empty list ignored, nothing to do}%
380 }
381 {\loi_def#1[#1]{\expanded{\expandafter\readlist_d\expanded{\loi_normalizeindex

```

```

382     {#3}{##1}{#3}}%
\loi_argcsname\loi_def{#3sep}[##1]{\expanded{\expandafter\readlist_d\expanded{\↵
    loi_normalizeindex{#3}{##1}{#3sep}}}%
383 \readlist_e{#2}%
384 \loi_argcsname\loi_argcsname\loi_let{#3len}{#3len[0]}% longueur du niveau 0
385 }%
386 }
387 \def\readlist_d#1#2#3{%
388 \unexpanded\expandafter\expandafter\expandafter{\csname#3[#2]\expandafter\endcsname\↵
    expandafter}%
389 \expanded{\loi_ifempty{#1}{\unexpanded{\unexpanded{\loi_error{#1}}}}}%
390 }
391 \def\readlist_e{%
392 \loi_argcsname\loi_let\loi_currentsep\loi_listofsep[number\loi_nestcnt]}%
393 \expandafter\readlist_f\loi_currentsep|\_nil
394 }
395 \long\def\readlist_f#1| |#2\_nil#3{\readlist_g1{#3#1}}% #1=<sep courant simple> #3=liste -> ↵
    rajoute un élément vide pour le test \ifempty ci dessous
396 \long\def\readlist_g#1#2{% #1=compteur d'index #2=liste d'éléments à examiner terminée par <↵
    sep courant simple> >>RIEN laissé après
397 \loi_ifempty{#2}
398 {\loi_argcsname\loi_edef{\loi_listname len[\csname loi_previndex[number\loi_nestcnt]\↵
    endcsname0]}{\number\numexpr#1-1\relax}%
399 \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
400 \advance\loi_nestcnt-1
401 \loi_argcsname\loi_let\loi_currentsep{\loi_listofsep[number\loi_nestcnt]}%
402 }
403 {\loi_expafter{\readlist_h{#2}{}}\loi_currentsep|\_loi_quark|#2\_nil{#1}% aller isoler le ↵
    ler item
404 }%
405 }
406 \long\def\readlist_h#1#2#3| |{% #1=liste restante #2=<dernier sep utilisé> #3=<sep courant>
407 \loi_ifx{\loi_quark#3}% on a épuisé tous les <séparateurs> ? RESTE à lire <expr+sep1>\_nil{<↵
    compteur>}
408 {\loi_ifempty{#2}% si #2 vide, aucun <sep utilisé> n'a été trouvé, il reste à lire "<liste ↵
    complète>\_nil"
409 {\long\def\readlist_i##1\_nil##2{\loi_exparg{\readlist_j{##2}{}}{\loi_gobarg##1}{#2}}% ↵
    ##2=compteur d'index
410 }
411 {\loi_ifx{\loi_listofpair\empty}% paires définies ?
412 {\long\def\readlist_i##1#2##2\_nil##3{\loi_exparg{\readlist_j{##3}{##2}}{\loi_gobarg ↵
    ##1}{#2}}%
413 }
414 {\long\def\readlist_i##1\_nil##2{%
415 \loi_exparg{\loi_exparg\loi_grabpaired_expr\loi_listofpair}{\loi_gobarg##1}{#2}\↵
    loi_grabpaired_result\loi_grabpaired_remain
416 \loi_exparg{\loi_exparg{\readlist_j{##2}}{\loi_grabpaired_remain}}{\↵
    loi_grabpaired_result}{#2}}%
417 }%
418 }%
419 \readlist_i\relax% le \relax meuble l'argument délimité
420 }
421 {\long\def\readlist_i##1#3##2\_nil{%
422 \loi_ifempty{##2}% si <liste restante> ne contient pas le <sep courant>
423 {\readlist_h{#1}{#2}% recommencer avec le même <sep utile>
424 }%
425 {\loi_ifx{\loi_listofpair\empty}% si pas de paires définies
426 {\loi_exparg\readlist_h{\loi_gobarg##1#3}{#3}% raccourcir <liste restante> et <sep ↵
    courant>:=<sep utile>% ##BUGFIX v1.53
427 }%

```

```

428     {\loi_exparg\loi_grabpaired_expr\loi_listofpair{#1}{#3}\loi_grabpaired_result\
      loi_grabpaired_remain
429     \loi_exparg\readlist_h{\loi_grabpaired_result#3}{#3}%
430     }%
431   }%
432 }%
433 \readlist_i\relax#1#3\_nil% ##BUGFIX v1.53
434 }%
435 }
436 \long\def\readlist_j#1#2#3{% #1=compteur d'index #2=liste restante #3=élément courant
437 \loi_ifnum{0\loi_exparg\loi_ifspacefirst{\loi_currentsep}{1\if_removeextremespaces1\fi=11 }%
      s'il faut retirer les espaces extrêmes
438 {\loi_exparg{\loi_exparg\readlist_k{#1}{#2}}{\loi_removeextremespaces{#3}}% redéfinir l'
      élément courant
439 {\readlist_k{#1}{#2}{#3}}%
440 }
441 \long\def\readlist_k#1#2#3#4{% #1=compteur d'index #2=liste restante #3=élément courant #4=
      sep utilisé
442 \loi_ifnum{0\if_ignoreemptyitems1\fi\loi_ifempty{#3}1{}=11 }
443 {\readlist_g{#1}{#2}% si l'on n'ignore pas les éléments vides
444 }%
445 {\loi_argcsname\loi_def{\loi_listname[\csname loi_previndex[\number\loi_nestcnt]\endcsname\
      #1]}{#3}% assignation de l'item ctuel à la macro
446 \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
447 \loi_ifnum{\loi_nestcnt<\nestdepth\relax}% si imbrication max non atteinte
448 {\advance\loi_nestcnt1
449 \loi_argcsname\edef\loi_previndex[\number\loi_nestcnt]}{\csname loi_previndex[\number\
      numexpr\loi_nestcnt-1]\endcsname#1,%
450 \readlist_e{#3}% recommencer avec l'élément courant
451 }
452 }%
453 \loi_exparg\readlist_g{\number\numexpr#1+1}{#2}% puis chercher l'élément suivant dans la
      liste restante
454 }%
455 }
456
457 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
458 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macro \listlen %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
459 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
460 \def\listlen#1[#2]{%
461 \expanded{%
462 \loi_ifempty{#2}
463 {\csname\loi_macroname#1len[0]\endcsname}
464 {\loi_exparg\listlen_a{\expanded{\loi_macroname#1}}{#2}}%
465 }%
466 }
467 \def\listlen_a#1#2{% #1=macro name #2=index non normalisé prendre <profondeur max-1>
468 \expandafter\listlen_b\expanded{\loi_normalizeindex{#1}{#2}}{#1}%
469 }
470 \def\listlen_b#1#2#3{% #1=err #2=index normalisé #3=macroname
471 \csname#3len[#2,0]\expandafter\endcsname
472 \expanded{\loi_ifempty{#1}}{\unexpanded{\unexpanded{\loi_error{#1}}}}%
473 }
474
475 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
476 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macro \foreachitem %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
477 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
478 \def\foreachitem#1\in#2{%
479 \edef\foreachitem_a{\noexpand\foreachitem_c\noexpand#1{\expandafter\noexpand\csname\
      loi_macroname#1cnt\endcsname}{\loi_macroname#2}}%

```

```

480 \futurelet\loi_nxttok\foreachitem_b
481 }
482 \def\foreachitem_b{\loi_ifx{\loi_nxttok[]}\foreachitem_a{\foreachitem_a[]}}
483 \def\foreachitem_c#1#2#3[#4]{% prendre <profondeur max-1>
484 \expandafter\foreachitem_d\expanded{\loi_normalizeindex{#3}{#4}}#1{#2}{#3}%
485 }
486 \def\foreachitem_d#1#2{\loi_ifempty{#2}{\foreachitem_e{#1}{}}{\foreachitem_e{#1}{#2}}}% #1=err ↵
487 #2=index norm
488 \long\def\foreachitem_e#1#2#3#4#5#6{% #1=err #2=index norm #3=macroiter #4=compteur associé ↵
489 #5=nom de macrolist #6=code
488 \loi_ifnum{\csname#5len[#20]\endcsname>0 }
489 {\loi_ifempty{#1}{}\loi_error{#1}}%
490 \loi_fornum#4=1to\csname#5len[#20]\endcsname\do{\loi_argcsname\let#3{#5[#2#4]}#6}%
491 }
492 {}%
493 }
494
495 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
496 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macro \showitem %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
497 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
498 \def\showitems{\loi_ifstar{\let\showitems_cmd\detokenize\showitems_a}{\let\showitems_cmd\
499 loi_identity\showitems_a}}
500 \def\showitems_a#1{\def\showitems_b{\showitems_d#1}\futurelet\loi_nxttok\showitems_c}
501 \def\showitems_c{\loi_ifx{\loi_nxttok[]}\showitems_b{\showitems_b[]}}
502 \def\showitems_d#1[#2]{\foreachitem\showitems_iterin#1[#2]{\showitemsmacro{\expandafter\
503 showitems_cmd\expandafter{\showitems_iter}}}}
504 \unless\ifdefined\fbbox
505 \newdimen\fbboxrule \newdimen\fbboxsep \fbboxrule=.4pt \fbboxsep=3pt % réglages identiques à ↵
506 LaTeX
507 \def\fbbox#1{% imitation de la macro \fbbox de LaTeX, voir pages 271 à 274 de "Apprendre à ↵
508 programmer en TeX"
509 \hbox{%
510 \vrule width\fbboxrule
511 \vtop{%
512 \vbox{\hrule height\fbboxrule \kern\fbboxsep \hbox{\kern\fbboxsep#1\kern\fbboxsep}}%
513 \kern\fbboxsep \hrule height\fbboxrule
514 } \vrule width\fbboxrule
515 }%
516 }
517 \fi
518 \def\showitemsmacro#1{% encadrement par défaut
519 \begingroup\fbboxsep=0.25pt \fbboxrule=0.5pt \fbbox{\strut#1}\endgroup
520 \hskip0.25em\relax
521 }
522 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
523 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% macro \itemtomacro %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
524 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
525 \def\itemtomacro#1[#2]{% #1[#2]=item non encore lu: #3=macro
526 \edef\loi_listname{\loi_macroname#1}%
527 \expandafter\itemtomacro_a\expanded{\loi_normalizeindex{\loi_listname}{#2}}\let
528 }
529 \def\gitemtomacro#1[#2]{% #1[#2]=item
530 \xdef\loi_listname{\loi_macroname#1}%
531 \expandafter\itemtomacro_a\expanded{\loi_normalizeindex{\loi_listname}{#2}}{\global\let}%
532 }
533 \def\itemtomacro_a#1#2#3#4{%
534 \loi_ifempty{#1}{}\loi_error{#1}}%
535 \loi_argcsname#3#4{\loi_listname[#2]}%
536 }

```



```

535 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
536 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% réglages par défaut %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
537 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
538 \newif\if_removeextremespaces
539 \newif\if_ignoreemptyitems
540 \let\ignoreemptyitems\_ignoreemptyitemstrue
541 \let\reademptyitems\_ignoreemptyitemsfalse
542 \loi_def_foreachsep{,}
543 \loi_restorecatcode
544 \reademptyitems
545 \setsepchar{,}
546 \defpair{}
547 \endinput
548
549 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
550 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% historique %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
551 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
552 v1.0    19/8/2016
553 - Première version publique
554 -----
555 v1.1    01/09/2016
556 - Stockage des séparateurs dans <macro>sep
557 - bug corrigé dans \loi_restorecatcode
558 -----
559 v1.2    22/10/2016
560 - macros \greadlist et \gitemtomacro pour la globalité
561 -----
562 v1.3    18/11/2016
563 - bugs corrigés dans la gestion de la globalité
564 -----
565 v1.4    05/10/2017
566 - test \loi_ifprimitive ajouté au test \loi_ifcs
567 - suppression de \loi_expafternil, création de \loi_expafter,
568   modification de \loi_argcsname
569 - correction d'un bug : \setsepchar{\par} ne provoque plus d'erreur.
570   \loi_ifnum devient \long
571 -----
572 v1.5    06/10/2017
573 - correction d'un bug dans \loi_ifcs
574 -----
575 v1.51   24/10/2017
576 - correction d'un bug dans \loi_ifcs
577 -----
578 v1.52   13/01/2018
579 - le dernier séparateur est <vide>
580 -----
581 v1.53   13/03/2018
582 - correction d'un bug dans \readlist_i
583 -----
584 v1.6    01/11/2018
585 - possibilité d'appariement de tokens dans les items
586 -----
587 v1.61   03/03/2019
588 - la macro \loi_ifcs contient une erreur de conception. Il faut
589   tester si le token est un sc && s'il est développable pour
590   renvoyer vrai car il existe des sc non développables && qui ne
591   sont pas des primitives.
592   Macro rebaptisée \loi_ifcsexpandable
593 -----
594 v1.62   18/05/2019
595 - utilisation de la nouvelle primitive \expanded au lieu du

```

596 désormais obsolète `\romannumeral`
597 - bug corrigé dans `\loi_ifcsexpandable`