The simples-matrices package*

Yvon Henel†

Released 2022/06/19

Résumé

Cette extension fournit des commandes pour définir et écrire des matrices en donnant leurs coefficients, par ligne, dans une vliste (liste de valeurs séparées par des virgules).
La documentation française pour l'utilisateur de l'extension simples-matrices est disponible sous le nom de `simples-matrices-fra`.

Abstract

This package provides macros to define and write matrices the coefficients of which are given, row by row, in a clist (list of values separated by commas).
The English documentation for the final user of the package simples-matrices is available in the file `simples-matrices-eng`.

1 Summary of Syntax of Document Commands

`\matrice` `\matrice(`⟨*prefix*⟩`) < `⟨*clist of key-value pairs*⟩` > [`⟨*type*⟩`]{`⟨*clist of coefficients*⟩`}`

`\declarermatrice` `\declarermatrice*(`⟨*prefix*⟩`) < `⟨*clist of key-value pairs*⟩` > {`⟨*matrix name*⟩`}[`⟨*type*⟩`]{`⟨*clist of coefficients*⟩`}`

`\lamatrice` `\lamatrice(`⟨*prefix*⟩`) < `⟨*clist of key-value pairs*⟩` > {`⟨*matrix name*⟩`}`

`\MatriceInterieur` `\MatriceInterieur`

`\LaMatriceInterieur` `\LaMatriceInterieur{`⟨*matrix name*⟩`}`

*This file describes v1.0, last revised 2022/06/19.

†E-mail: le.texnicien.de.surface@yvon-henel.fr

\simplesmatricessetup \simplesmatricessetup{⟨*clist of key-value pairs*⟩}

\matid \matid[⟨*diagonal coefficient, default* 1⟩]{⟨*size of the square matrix*⟩}

\matnulle \matid[⟨*diagonal coefficient, default* 0⟩]{⟨*size of the square matrix*⟩}

2 simples-matrices implementation

2.1 Loading And Greeting

```
1 ⟨@@=SMPLMTRC⟩
2 \RequirePackage{l3keys2e,xparse}
3 \ProvidesExplPackage
4 {simples-matrices} {2022/06/08} {1.0} {defining and printing matrices}
5 \RequirePackage{amsmath}
```

amsmath is the only auxiliary package loaded. We need it to write the matrices.

2.2 Keys And Options

I'm using the key management mechanism of LaTeX3.

```
6 \keys_define:nn {simples-matrices}
7 {
8   envir .tl_set:N = \g__SMPLMTRC_nom_environnement_tl,
9   envir .initial:n = {matrix},
10  prefix .tl_set:N = \g__SMPLMTRC_prefix_environnement_tl,
11  prefix .initial:n = {p},
12  argopt .tl_set:N = \g__SMPLMTRC_argument_environnement_tl,
13  argopt .initial:n = { },
14  typeord .str_set:N = \g__SMPLMTRC_type_ordinaire_str,
15  typeord .initial:n = { C },
16 }
17 \ProcessKeysOptions {simples-matrices}
```

The keys above defined the package options. The next one may be usefull to return to a kind of neutral state.

```
18 \keys_define:nn {simples-matrices}
19 {
20   out-of-box .meta:n = {prefix=p, envir=matrix, argopt=, typeord=C},
21 }
```

2.3 Constants And Variables

To handle the syntax of `xcas` matrices, I resort to *regex* so I have to create some constant *regexes*. Some comments are just there to prevent `emacs` to go too far in formating the code.

```
22 \regex_const:Nn \c__SMPLMTRC_xcas_debut_regexp {\[\[} %\]\]
23 \regex_const:Nn \c__SMPLMTRC_xcas_fin_regexp {\]\]}
24 \regex_const:Nn \c__SMPLMTRC_xcas_milieu_regexp{\],\ *\[} %\]
```

Now come error messages: the first deals with bad numbers of coefficient for matrices of type I, J, S or T; the second appears when dealing with an x type matrix.

```
\msg_new:nnnn{simples-matrices}{errorIJST}
{The~number~of~coefficients~does~not~fit~the~matrix-type}
{I~was~expecting~#1~coefficients~and~obtain~only~#2.}

\msg_new:nnnn{simples-matrices}{errorx}
{The~data~do~not~seem~to~match~the~x-type}
{I~was~expecting~the~data~to~begin~with~"[["}
```

To contain the inner part of the matrix, we need a token list global variable. I chose to built it with a non-letter to try to prevent name clash.

```
\tl_new:c {g__SMPLMTRC_contenu_anonyme*_tl}
```

__SMPLMTRC_creer_matrice_aux: When this macro is called, `\g_tmpa_int` contains the number of columns of the matrix and `\g_tmpa_seq` the *sequence* of coefficients.

```
\cs_new:Nn \__SMPLMTRC_creer_matrice_aux:
{
  \tl_gclear:c {g__SMPLMTRC_contenu_anonyme*_tl}

  \seq_map_indexed_inline:Nn \g_tmpa_seq
  {
    \tl_gput_right:cn {g__SMPLMTRC_contenu_anonyme*_tl} { ##2  }
    \int_compare:nNnTF { \int_mod:nn {##1} { \g_tmpa_int } } = { 0 }
    {
      \tl_gput_right:cn {g__SMPLMTRC_contenu_anonyme*_tl} { \\ }
    }
    {
      \tl_gput_right:cn {g__SMPLMTRC_contenu_anonyme*_tl} { &  }
    }
  }
}
```

At the end of this macro, `\g__SMPLMTRC_contenu_anonyme*_tl` contains the inner part of the matrix.

(*End definition for* `\__SMPLMTRC_creer_matrice_aux:`.)

For I need to match a text against a regex constant, I create a variant of one of the function of LaTeX3:

```
\cs_generate_variant:Nn \regex_match:nnF { nVF }
```

The next macro does the bulk of the work in reading the coefficients in the second argument and managing them according to the data-type which is given in the first argument.

__SMPLMTRC_creer_matrice:nn To be on the safe side, I begin with clearing the variable.

```
\cs_new:Nn \__SMPLMTRC_creer_matrice:nn
{
  \tl_gclear:c {g__SMPLMTRC_contenu_anonyme*_tl}
```

I then read the first argument inside `\l_tmpa_str` with `\str_set:Nx` in case the string is contained in a variable.

```
  \str_set:Nx \l_tmpa_str { #1 }
```

If `#1` is equal to `0`, I fetch the data-type in the string valued through the key `typeord`.

```
\str_if_eq:VnT \l_tmpa_str { 0 }
{
  \str_set:NV \l_tmpa_str \g__SMPLMTRC_type_ordinaire_str
}
```

We enter now the first `case` which depends on the data-type. The first case deals with the `x`-type. Here we obtain the inner part of the matrix by replacing, according to *regex*. There is no need to calculate anything.

If ever there are other types of the same kind —e. g. to deal with `maxima` syntax or the like— those types would be dealt with here.

```
\str_case:VnF \l_tmpa_str
{
  { x } {
    \tl_set:Nn \l_tmpa_tl { #2 }

    \regex_match:nVF {\A\s*\[\[} %\]\]
    \l_tmpa_tl
    {
        \msg_fatal:nn{simples-matrices} {errorx}
    }

    \regex_replace_all:NnN \c__SMPLMTRC_xcas_debut_regexp { } \l_tmpa_tl
    \regex_replace_all:NnN \c__SMPLMTRC_xcas_fin_regexp { \c{\\} } \l_tmpa_tl
    \regex_replace_all:NnN \c__SMPLMTRC_xcas_milieu_regexp { \c{\\} } \l_tmpa_tl
    \regex_replace_all:nnN { , }{ & } \l_tmpa_tl

    \tl_set_eq:cN {g__SMPLMTRC_contenu_anonyme*_tl} \l_tmpa_tl
  }
}
```

The `x` -type is done. Now for the `false` part of the `case` where I will take care of the other kind of data-types, those which require calculations.

I prepare the work by reading the second argument inside the *sequence* `\g_tmpa_seq` and cleaning `\g_tmpb_seq`.

```
{
  \seq_set_from_clist:Nn \g_tmpa_seq { #2 }
  \seq_clear:N \l_tmpb_seq
```

Now come the second `case`:

```
  \str_case:VnF \l_tmpa_str
  {
```

The first case is that of the square matrices. I calculate the number of columns as the square root of the number of coefficients. There is, as for now, no check for correctness because in case of bad number of coefficients there will just be a strange matrix, with void cells.

```
    { C } {
      \int_set:Nn \g_tmpb_int { \seq_count:N \g_tmpa_seq }
      \int_set:Nn \g_tmpa_int { \fp_to_int:n { sqrt(\g_tmpb_int) } }
    }
```

In case of a diagonal matrix, the number of columns is simply the number of coefficients but we have to create the suitable *sequence*. The trick is just to add as many zeros as there are columns between each given coefficient.

```
      { D } {
        \int_set:Nn \g_tmpa_int { \seq_count:N \g_tmpa_seq }
        \seq_pop_left:NN \g_tmpa_seq \l_tmpa_tl
        \seq_put_right:NV \l_tmpb_seq \l_tmpa_tl

        \bool_until_do:nn { \seq_if_empty_p:N \g_tmpa_seq }
        {
          \int_step_inline:nn { \g_tmpa_int }
          {
            \seq_put_right:Nn \l_tmpb_seq { 0 }
          }
          \seq_pop:NN \g_tmpa_seq \l_tmpa_tl
          \seq_put_right:NV \l_tmpb_seq \l_tmpa_tl
        }
        \seq_set_eq:NN \g_tmpa_seq \l_tmpb_seq
      }
```

The code for the next four cases is very similar. We have to calculate the number of columns from the number of coefficient and then complete the *sequence* of coefficients.

Here I check the correctness because, otherwise, in case of error LaTeX doesn't behave kindly.

The number of coefficients *non-necessarily* null in a triangular matrix with n columns is $n(n+1)/2$ or $n(n-1)/2$ if the diagonal is *null*. Therefore the computing of `\g_tmpa_int` from `\g_tmpb_int` which is equal to the number of coefficients, that is the number of items in the *sequence* `\g_tmpa_seq`.

In the first case (upper triangular) there are coefficients on the diagonal so we have to append $L-1$ zeros before $N+1-L$ coefficients where L is the line number and N the number of columns.

```
      { S } {
        \int_set:Nn \g_tmpb_int { \seq_count:N \g_tmpa_seq }
        \int_set:Nn \g_tmpa_int
        {
          \fp_to_int:n { ( sqrt(1 + 8*\g_tmpb_int) - 1 ) / 2 }
        }
```

We check for correctness and abort in case of discrepancy.

```
        \int_compare:nNnF
        {2*\g_tmpb_int} = { \g_tmpa_int * (\g_tmpa_int + 1)}
        {
          \msg_fatal:nnxx{simples-matrices}
          {errorIJST}
          {\int_to_arabic:n{(\g_tmpa_int * (\g_tmpa_int + 1))/2}}
          {\int_to_arabic:n{\g_tmpb_int}}
        }
```

`##1` is the line number of the matrix

```

        \int_step_inline:nn { \g_tmpa_int }
        {
          \int_step_inline:nn { ##1 - 1 }
          {
```

```
            \seq_put_right:Nn \l_tmpb_seq {0}
          }
          \int_step_inline:nn { \g_tmpa_int - ##1 + 1 }
          {
            \seq_pop:NN \g_tmpa_seq \l_tmpa_tl
            \seq_put_right:NV \l_tmpb_seq \l_tmpa_tl
          }
        }

        \seq_set_eq:NN \g_tmpa_seq \l_tmpb_seq

      }
```

When there are zeros on the diagonal, the formula changes a tiny bit: append L zeros before $N - L$ coefficients.

```
      { T } {
        \int_set:Nn \g_tmpb_int { \seq_count:N \g_tmpa_seq }
        \int_set:Nn \g_tmpa_int
        {
          \fp_to_int:n { (1 + sqrt(1 + 8*\g_tmpb_int) ) / 2 }
        }
        \int_compare:nNnF
        {2*\g_tmpb_int} = { \g_tmpa_int * (\g_tmpa_int - 1)}
        {
          \msg_fatal:nnxx{simples-matrices}
          {errorIJST}
          {\int_to_arabic:n{(\g_tmpa_int * (\g_tmpa_int - 1))/2}}
          {\int_to_arabic:n{\g_tmpb_int}}
        }
        \int_step_inline:nn { \g_tmpa_int }
        {
          \int_step_inline:nn { ##1 }
          {
            \seq_put_right:Nn \l_tmpb_seq {0}
          }
          \int_step_inline:nn { \g_tmpa_int - ##1 }
          {
            \seq_pop:NN \g_tmpa_seq \l_tmpa_tl
            \seq_put_right:NV \l_tmpb_seq \l_tmpa_tl
          }
        }
        \seq_set_eq:NN \g_tmpa_seq \l_tmpb_seq
      }
```

Now come the cases for lower triangular matrices. First *with diagonal*.

```
      { I } {
        \int_set:Nn \g_tmpb_int { \seq_count:N \g_tmpa_seq }
        \int_set:Nn \g_tmpa_int
        {
          \fp_to_int:n { ( sqrt(1 + 8*\g_tmpb_int) - 1 ) / 2 }
        }
        \int_compare:nNnF
        {2*\g_tmpb_int} = { \g_tmpa_int * (\g_tmpa_int + 1)}
        {
          \msg_fatal:nnxx{simples-matrices}
```

```
          {errorIJST}
          {\int_to_arabic:n{(\g_tmpa_int * (\g_tmpa_int + 1))/2}}
          {\int_to_arabic:n{\g_tmpb_int}}
        }
        \int_step_inline:nn { \g_tmpa_int }
        {
          \int_step_inline:nn { ##1 }
          {
            \seq_pop:NN \g_tmpa_seq \l_tmpa_tl
            \seq_put_right:NV \l_tmpb_seq \l_tmpa_tl
          }
          \int_step_inline:nn { \g_tmpa_int - ##1 }
          {
            \seq_put_right:Nn \l_tmpb_seq {0}
          }
        }
        \seq_set_eq:NN \g_tmpa_seq \l_tmpb_seq
      }
```

Secondly *without diagonal*.

```
      { J } {
        \int_set:Nn \g_tmpb_int { \seq_count:N \g_tmpa_seq }
        \int_set:Nn \g_tmpa_int
        {
          \fp_to_int:n { (1 + sqrt(1 + 8*\g_tmpb_int) ) / 2 }
        }
        \int_compare:nNnF
        {2*\g_tmpb_int} = { \g_tmpa_int * (\g_tmpa_int - 1)}
        {
          \msg_fatal:nnxx{simples-matrices}
          {errorIJST}
          {\int_to_arabic:n{(\g_tmpa_int * (\g_tmpa_int - 1))/2}}
          {\int_to_arabic:n{\g_tmpb_int}}
        }
        \int_step_inline:nn { \g_tmpa_int }
        {
          \int_step_inline:nn { ##1 -1}
          {
            \seq_pop:NN \g_tmpa_seq \l_tmpa_tl
            \seq_put_right:NV \l_tmpb_seq \l_tmpa_tl
          }
          \int_step_inline:nn { \g_tmpa_int - ##1 + 1}
          {
            \seq_put_right:Nn \l_tmpb_seq {0}
          }
        }
        \seq_set_eq:NN \g_tmpa_seq \l_tmpb_seq
      }
    }
```

Here we attain the `false` part of the second `case`. We assume that we have a number in `#1`.

```
    {
      \int_set:Nn \g_tmpa_int { \l_tmpa_str }
    }
```

If all is well, here it ends well with the number of columns in `\g_tmpa_int` and all the coefficients of the matrix in `\g_tmpa_seq`. So it's time to call

```
    \__SMPLMTRC_creer_matrice_aux:
  }
}
```

(*End definition for* `\__SMPLMTRC_creer_matrice:nn`.)

`\MatriceInterieur` A document command to access the inner part of the **last** printed —via `\matrice`— or defined —via `\declarermatrice`— matrix.

This macro can be placed inside an `array`-like environment.

```
\cs_new:Npn \MatriceInterieur
{
  \tl_use:c { g__SMPLMTRC_contenu_anonyme*_tl }
}
```

(*End definition for* `\MatriceInterieur`. *This function is documented on page* 1.)

`\LaMatriceInterieur` A version to access the inner part of a named matrix:

```
\cs_new:Npn \LaMatriceInterieur #1
{
  \tl_use:c { g__SMPLMTRC_contenu_nom_#1_tl }
}
```

(*End definition for* `\LaMatriceInterieur`. *This function is documented on page* 1.)

The next macro is used to print a matrix. Its name means *prepare (the) presentation* of the matrix. It builds the name of the environment, taking care of possible arguments of the said environment.

`\__SMPLMTRC_preparer_presentation:nn` At the end of the macro, `\g_tmpa_tl` contains the *overture* of the environment and `\g_tmpb_tl` the *grand final*.

```
\cs_new:Nn \__SMPLMTRC_preparer_presentation:nn
{
  \IfValueT{#1}
  {
    \keys_set:nn {simples-matrices} {prefix = #1}
  }
  \IfValueT{#2}
  {
    \keys_set:nn {simples-matrices} { #2 }
  }
  \tl_set:NV \l_tmpa_tl \g__SMPLMTRC_prefix_environnement_tl
  \tl_put_right:NV \l_tmpa_tl \g__SMPLMTRC_nom_environnement_tl
  \tl_set:Nn \g_tmpa_tl { \begin{\l_tmpa_tl} }
  \tl_put_right:NV \g_tmpa_tl \g__SMPLMTRC_argument_environnement_tl
  \tl_set:Nn \g_tmpb_tl {\end{\l_tmpa_tl} }
}
```

(*End definition for* `\__SMPLMTRC_preparer_presentation:nn`.)

Now comes the first main document command which prints the matrix from the data-type and coefficients. It assigns the different arguments to the auxiliary macros and finishes with using the token lists created by the auxiliaries.

\matrice

```
\NewDocumentCommand{\matrice}{ d() d<> O{0} m }
{
  \__SMPLMTRC_creer_matrice:nn { #3 } { #4 }
  \__SMPLMTRC_preparer_presentation:nn { #1 } { #2 }
  \tl_use:N \g_tmpa_tl
  \tl_use:c { g__SMPLMTRC_contenu_anonyme*_tl }
  \tl_use:N \g_tmpb_tl
}
```

(*End definition for* `\matrice`*. This function is documented on page* 1.)

I provide a way of changing the keys inside the document with the usual *setup* macro:

\simplesmatricessetup

```
\NewDocumentCommand{\simplesmatricessetup}{m}
{
  \keys_set:nn {simples-matrices} { #1 }
}
```

(*End definition for* `\simplesmatricessetup`*. This function is documented on page* 2.)

Another main document command to define a named matrix. The starred version defines and prints.

The first mandatory argument —the 4th of all— should contain the "name" of the matrix with which one can recall the matrix using `\lamatrice`.

\declarermatrice

```
\NewDocumentCommand{\declarermatrice}{ s d() d<> m  O{0} m }
{
  \__SMPLMTRC_creer_matrice:nn { #5 } { #6 }
  \tl_gset_eq:cc {g__SMPLMTRC_contenu_nom_#4_tl} {g__SMPLMTRC_contenu_anonyme*_tl}
  \IfBooleanT { #1 }
  {
    \__SMPLMTRC_preparer_presentation:nn { #2 } { #3 }
    \tl_use:N \g_tmpa_tl
    \tl_use:c { g__SMPLMTRC_contenu_anonyme*_tl }
    \tl_use:N \g_tmpb_tl
  }
}
```

(*End definition for* `\declarermatrice`*. This function is documented on page* 1.)

To print a named matrix:

\lamatrice

```
\NewDocumentCommand{\lamatrice}{ d() d<> m }
{
  \__SMPLMTRC_preparer_presentation:nn { #1 } { #2 }
  \tl_use:N \g_tmpa_tl
  \tl_use:c { g__SMPLMTRC_contenu_nom_#3_tl }
  \tl_use:N \g_tmpb_tl
}
```

(*End definition for* `\lamatrice`*. This function is documented on page* 1.)

For the next commands, I need a variant:

```
\cs_generate_variant:Nn \__SMPLMTRC_creer_matrice:nn { nV }
```

The auxiliary macro to print an identity matrix. #1 will receive the value to print in the diagonal —defaulting to 1— and #2 the number of columns.

__SMPLMTRC_creer_matrice_identite:nn

```
\cs_new:Nn \__SMPLMTRC_creer_matrice_identite:nn
{
  \clist_clear:N \l_tmpa_clist
  \int_step_inline:nn { #2 }
  {
    \clist_put_left:Nn \l_tmpa_clist { #1 }
  }
  \__SMPLMTRC_creer_matrice:nV {D} \l_tmpa_clist
}
```

(*End definition for* `\__SMPLMTRC_creer_matrice_identite:nn`.)

I can then use it to define the document command:

\matid

```
\NewDocumentCommand{\matid}{ d() d<> O{1} m}
{
  \__SMPLMTRC_creer_matrice_identite:nn { #3 } { #4 }
  \__SMPLMTRC_preparer_presentation:nn  { #1 } { #2 }
  \tl_use:N \g_tmpa_tl
  \tl_use:c { g__SMPLMTRC_contenu_anonyme*_tl }
  \tl_use:N \g_tmpb_tl
}
```

(*End definition for* `\matid`. *This function is documented on page* 2.)

I follow the same pattern for null matrices. First the auxiliary macro, #2 is the number of columns, #1 the value of all the coefficients —defaulting to 0—.

__SMPLMTRC_creer_matrice_nulle:nn

```
\cs_new:Nn \__SMPLMTRC_creer_matrice_nulle:nn
{
  \clist_clear:N \l_tmpa_clist
  \int_step_inline:nn { #2*#2 }
  {
    \clist_put_left:Nn \l_tmpa_clist { #1 }
  }
  \__SMPLMTRC_creer_matrice:nV {C} \l_tmpa_clist
}
```

(*End definition for* `\__SMPLMTRC_creer_matrice_nulle:nn`.)

I can again use it to define the document command:

\matnulle

```
\NewDocumentCommand{\matnulle}{ d() d<> O{0} m}
{
  \__SMPLMTRC_creer_matrice_nulle:nn {#3} {#4}
  \__SMPLMTRC_preparer_presentation:nn { #1 } { #2 }
  \tl_use:N \g_tmpa_tl
  \tl_use:c { g__SMPLMTRC_contenu_anonyme*_tl }
  \tl_use:N \g_tmpb_tl
}
```

(*End definition for* `\matnulle`. *This function is documented on page* 2.)

Change History

v1.0 *version publique.* 1
General: First public version.*Première*

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

\[. 22, 24, 63
\\ 42, 70, 71
\␣ 24
\] 22, 23, 24, 63

A

\A 63

B

\begin 244
bool commands:
\bool_until_do:nn 91

C

\c 70, 71
clist commands:
\clist_clear:N 282, 299
\clist_put_left:Nn 285, 302
\l_tmpa_clist 282, 285, 287, 299, 302, 304
cs commands:
\cs_generate_variant:Nn 49, 279
\cs_new:Nn 33, 50, 232, 280, 297
\cs_new:Npn 224, 228

D

\declarermatrice *1*, *8*, 260
\declarermatrice* *1*

E

\end 246

F

fp commands:
\fp_to_int:n . . . 84, 106, 137, 165, 193

I

\IfBooleanT 264
\IfValueT 234, 238
int commands:
\int_compare:nNnTF 40, 108, 139, 167, 195
\int_mod:nn 40
\int_set:Nn 83, 84, 87, 103, 104, 134, 135, 162, 163, 190, 191, 219
\int_step_inline:nn 93, 117, 119, 123, 147, 149, 153, 175, 177, 182, 203, 205, 210, 283, 300
\int_to_arabic:n 113, 114, 144, 145, 172, 173, 200, 201
\g_tmpa_int *3*, *5*, *8*, 40, 84, 87, 93, 104, 109, 113, 117, 123, 135, 140, 144, 147, 153, 163, 168, 172, 175, 182, 191, 196, 200, 203, 210, 219
\g_tmpb_int *5*, 83, 84, 103, 106, 109, 114, 134, 137, 140, 145, 162, 165, 168, 173, 190, 193, 196, 201

K

keys commands:
\keys_define:nn 6, 18
\keys_set:nn 236, 240, 258

L

\lamatrice *1*, *9*, 272
\LaMatriceInterieur *1*, 228

M

\matid *2*, 289
\matnulle *2*, 306
\matrice *1*, *8*, 248
\MatriceInterieur *1*, 224
msg commands:
\msg_fatal:nn 66
\msg_fatal:nnnn . . . 111, 142, 170, 198
\msg_new:nnnn 25, 29

N

\NewDocumentCommand 248, 256, 260, 272, 289, 306

P

\ProcessKeysOptions 17
\ProvidesExplPackage 3

R

regex commands:
\regex_const:Nn 22, 23, 24
\regex_match:nnTF 49, 63
\regex_replace_all:NnN 69, 70, 71
\regex_replace_all:nnN 72
\RequirePackage 2, 5

S

\s 63
seq commands:
\seq_clear:N 79
\seq_count:N 83, 87, 103, 134, 162, 190
\seq_if_empty_p:N 91
\seq_map_indexed_inline:Nn 37
\seq_pop:NN 97, 125, 155, 179, 207
\seq_pop_left:NN 88
\seq_put_right:Nn 89, 95, 98, 121, 126, 151, 156, 180, 184, 208, 212
\seq_set_eq:NN 100, 130, 159, 187, 215
\seq_set_from_clist:Nn 78
\g_tmpa_seq . *3–5*, *8*, 37, 78, 83, 87, 88, 91, 97, 100, 103, 125, 130, 134, 155, 159, 162, 179, 187, 190, 207, 215
\g_tmpb_seq *4*
\l_tmpb_seq 79, 89, 95, 98, 100, 121, 126, 130, 151, 156, 159, 180, 184, 187, 208, 212, 215
\simplesmatricessetup *2*, 256
SMPLMTRC internal commands:
\g__SMPLMTRC_argument_environnement_tl 12, 245
\g__SMPLMTRC_contenu_anonyme*_tl . *3*
__SMPLMTRC_creer_matrice:nn 50, 50, 250, 262, 279, 287, 304
__SMPLMTRC_creer_matrice_aux: 33, 33, 221
__SMPLMTRC_creer_matrice_identite:nn 280, 280, 291
__SMPLMTRC_creer_matrice_nulle:nn 297, 297, 308
\g__SMPLMTRC_nom_environnement_tl 8, 243
\g__SMPLMTRC_prefix_environnement_tl 10, 242
__SMPLMTRC_preparer_presentation:nn 232, 232, 251, 266, 274, 292, 309
\g__SMPLMTRC_type_ordinaire_str 14, 56
\c__SMPLMTRC_xcas_debut_regexp 22, 69
\c__SMPLMTRC_xcas_fin_regexp 23, 70
\c__SMPLMTRC_xcas_milieu_regexp 24, 71
str commands:
\str_case:nnTF 58, 80
\str_if_eq:nnTF 54
\str_set:Nn *3*, 53, 56
\l_tmpa_str . . *3*, 53, 54, 56, 58, 80, 219

T

tl commands:
\tl_gclear:N 35, 52
\tl_gput_right:Nn 39, 42, 45
\tl_gset_eq:NN 263
\tl_new:N 32
\tl_put_right:Nn 243, 245
\tl_set:Nn 61, 242, 244, 246
\tl_set_eq:NN 74
\tl_use:N 226, 230, 252, 253, 254, 267, 268, 269, 275, 276, 277, 293, 294, 295, 310, 311, 312
\g_tmpa_tl *8*, 244, 245, 252, 267, 275, 293, 310
\l_tmpa_tl 61, 64, 69, 70, 71, 72, 74, 88, 89, 97, 98, 125, 126, 155, 156, 179, 180, 207, 208, 242, 243, 244, 246
\g_tmpb_tl *8*, 246, 254, 269, 277, 295, 312