A LaTeX FIFO/Stack implementation for Package writers

Stephen Gaito

2014/03/24

Abstract

This package provides a LaTeX implementation of a FIFO/Stack system independent from the TeX stack itself. It is based on Benjamin Bayart's excellent stack package from CTAN.

This package by itself is of no use to anyone other than macro-developers.

Note that to simplify the computations, this package makes essential use of ε-TeX's `\numexpr` macro. Fortunately most modern TeX, LaTeX and pdfLaTeX's are actually ε-TeX.

Contents

1 Using the FIFO/stack package

All of the macros provided by this package are meant to be used by package developers (there is no meaningful use of a FIFO or stack in a final document, they will only to be used in a package), so, all of those macros have capital letters in their names, following old guidelines for LaTeX3 project.

To ensure these macros do not clash with other packages, the names of all macros begin with "FS".

1.1 FIFO/Stack creation and destruction commands

To create a new FIFO/stack:

`\FSCreate` `\FSCreate{fifoStackName}{defaultValue}`

This macro simply initializes all the required data for the FIFO/stack named by the value of the first argument, `fifoStackName`. The second argument, `defaultValue`, is the value of the top of the FIFO/stack whenever the FIFO/stack is empty.

To clear an existing FIFO/stack of all of its current elements:

`\FSClear` `\FSClear{fifoStackName}`

This macro empties the FIFO/stack named `fifoStackName`, but the FIFO/stack can still be used to `\FSPush` or `\FSUnshift` new elements.

To destroy an existing FIFO/stack:

`\FSDestroy` `\FSDestroy{fifoStackName}`

This macro destroys the FIFO/stack named `fifoStackName`.

1.2 Stack based commands

To push a new value on *top* of a FIFO/stack:

`\FSPush` `\FSPush{fifoStackName}{value}`

Note that you *can* use `\FSTop` in the `value` argument of the `\FSPush` macro. See the qstest unit test for `\FSPush` (below) for an example.

To remove the value from the *top* of a FIFO/stack:

`\FSPop` `\FSPop{fifoStackName}`

To get the value from the *top* of the FIFO/stack:

`\FSTop` `\FSTop{fifoStackName}`

For use when debugging your use of the FIFO/stack:

`\FSShowTop` `\FSShowTop{fifoStackName}`

will `\typeout` the current value on the *top* of the FIFO/stack.

1.3 FIFO based commands

To add a new value to the *bottom* of a FIFO/stack:

`\FSUnshift` `\FSUnshift{fifoStackName}{value}`

Note that you *can* use `\FSBottom` in the `value` argument of the `\FSUnshift` macro. See the qstest unit test for `\FSUnshift` (below) for an example.

To remove the value on the *bottom* of a FIFO/stack:

`\FSShift` `\FSShift{fifoStackName}`

To get the value on the *bottom* of a FIFO/stack:

`\FSBottom` `\FSBottom{fifoStackName}`

For use when debugging your use of the FIFO/stack:

`\FSShowBottom` `\FSShowBottom{fifoStackName}`

will `\typeout` the current value on the *bottom* of the FIFO/stack.

1.4 FIFO/stack size commands

To get the current size of the FIFO/stack:

`\FSSize` `\FSSize{fifoStackName}`

1.5 TeXStudio CWL file

As a bonus for users of TeXStudio, the `fifo-stack.zip` file also contains a `fifo-stack.cwl` file which provides TeXStudio command expansion descriptions for each of the 'public' `fifo-stack` macros. To make use of these command expansions, simply place the `fifo-stack.cwl` file into TeXStudio's local configuration directory (on most Linux distributions this can be found in `~/.config/texstudio`).

This `fifo-stack.cwl` file is automatically generated from the macrocode lines which begin with the characters `%txs`. This automatic generation is done by the `cook texStyle` command defined by the sister project diSimplexLaTeX's (ruby) cookbook.

1.6 TEX and LATEX Dependencies

The `fifo-stack` package depends upon both ε-TEX, as well as the `ifthen` package. ε-TEX is required only for the two uses of `\numexpr`. The `ifthen` package is only required for the one use of `\whiledo`. All three uses are listed in the index.

The production of this documentation assumes you have the `hyperref` package installed.

The regression tests associated with this package depend upon the `qstest`, `ifthen`, and `xifthen` packages.

2 Integration tests and example usage

We use the QSTest package from CTAN to provide both integration and unit tests of our FIFO/stack package.

In this section we walk through a number of integration tests which also provide example uses of FIFO/Stacks. We provide unit tests of specific invariants in the code section below. These unit tests are associated with the section of code which implements a particular invariant.

We begin by setting up the LaTeX QSTest package to test the `fifo-stack` package, and we will log everything. Note that we do not use a document class or begin/end a document, this is because there should not be any *normal* output created. All *results* are listed in the associated `fifo-stack-test.lgout` file.

```
⟨*qstest⟩
\RequirePackage{qstest}
\RequirePackage{xifthen}
\IncludeTests{*}
\LogTests{lgout}{*}{*}
\RequirePackage{fifo-stack}
⟨/qstest⟩
```

2.1 Stack based examples

We now provide an example of standard use of just the stack based functionality. To do this we create a stack, push three things onto the stack and then pop all three things back off, all in a nice linear fashion.

```
⟨*qstest⟩
\begin{qstest}{Simple stack integration test}
  {\FSCreate, \FSPush, \FSPop, \FSTop}

  \FSCreate{testStack}{defaultValue}
  \Expect*{\FSTop{testStack}}{defaultValue}
  \FSPush{testStack}{firstValue}
  \Expect*{\FSTop{testStack}}{firstValue}
  \FSPush{testStack}{secondValue}
  \Expect*{\FSTop{testStack}}{secondValue}
  \FSPush{testStack}{thirdValue}
  \Expect*{\FSTop{testStack}}{thirdValue}
  \FSPop{testStack}
  \Expect*{\FSTop{testStack}}{secondValue}
  \FSPop{testStack}
  \Expect*{\FSTop{testStack}}{firstValue}
  \FSPop{testStack}
  \Expect*{\FSTop{testStack}}{defaultValue}
  \FSPop{testStack}
  \Expect*{\FSTop{testStack}}{defaultValue}
\end{qstest}
⟨/qstest⟩
```

We now look at using the stack in a non-linear order pushing, popping, then pushing and popping again.

```
⟨*qstest⟩
\begin{qstest}{Up down up down stack integration test}
  {\FSCreate, \FSPush, \FSPop, \FSTop}

  \FSCreate{testStackUDUD}{defaultValue}
  \Expect*{\FSTop{testStackUDUD}}{defaultValue}
  \FSPush{testStackUDUD}{firstValue}
  \Expect*{\FSTop{testStackUDUD}}{firstValue}
  \FSPush{testStackUDUD}{secondValue}
  \Expect*{\FSTop{testStackUDUD}}{secondValue}
  \FSPop{testStackUDUD}
  \Expect*{\FSTop{testStackUDUD}}{firstValue}
  \FSPush{testStackUDUD}{newSecondValue}
  \Expect*{\FSTop{testStackUDUD}}{newSecondValue}
\end{qstest}
⟨/qstest⟩
```

2.2 FIFO/Stack based examples

This time we make use of `\FSPush` to put items onto the FIFO and `\FSShift` to take them off in a *first in first off* order. At the end we shift past the top of the FIFO/Stack, and so we should get the default value again.

```
⟨*qstest⟩
\begin{qstest}{Simple FIFO integration test}
  {\FSCreate, \FSPush, \FSShift, \FSTop, \FSBottom}

  \FSCreate{testFifo}{defaultValue}
  \Expect*{\FSTop{testFifo}}{defaultValue}
  \Expect*{\FSBottom{testFifo}}{defaultValue}
  \FSPush{testFifo}{firstValue}
  \Expect*{\FSTop{testFifo}}{firstValue}
  \Expect*{\FSBottom{testFifo}}{firstValue}
  \FSPush{testFifo}{secondValue}
  \Expect*{\FSTop{testFifo}}{secondValue}
  \Expect*{\FSBottom{testFifo}}{firstValue}
  \FSShift{testFifo}
  \Expect*{\FSTop{testFifo}}{secondValue}
  \Expect*{\FSBottom{testFifo}}{secondValue}
  \FSPush{testFifo}{thirdValue}
  \Expect*{\FSTop{testFifo}}{thirdValue}
  \Expect*{\FSBottom{testFifo}}{secondValue}
  \FSShift{testFifo}
  \Expect*{\FSTop{testFifo}}{thirdValue}
  \Expect*{\FSBottom{testFifo}}{thirdValue}
  \FSShift{testFifo}
  \Expect*{\FSTop{testFifo}}{defaultValue}
```

```
  \Expect*{\FSBottom{testFifo}}{defaultValue}
  \FSShift{testFifo}
  \Expect*{\FSTop{testFifo}}{defaultValue}
  \Expect*{\FSBottom{testFifo}}{defaultValue}
\end{qstest}
```
⟨/qstest⟩

We now provide an example use of `\FSUnshift` and `\FSShift`.

```
⟨*qstest⟩
\begin{qstest}{Simple linear use of unshift/shift}
  {\FSUnshift, \FSShift}

  \FSCreate{testRStack}{defaultValue}
  \Expect*{\FSBottom{testRStack}}{defaultValue}
  \FSUnshift{testRStack}{firstValue}
  \Expect*{\FSBottom{testRStack}}{firstValue}
  \FSUnshift{testRStack}{secondValue}
  \Expect*{\FSBottom{testRStack}}{secondValue}
  \FSUnshift{testRStack}{thirdValue}
  \Expect*{\FSBottom{testRStack}}{thirdValue}
  \FSShift{testRStack}
  \Expect*{\FSBottom{testRStack}}{secondValue}
  \FSShift{testRStack}
  \Expect*{\FSBottom{testRStack}}{firstValue}
  \FSShift{testRStack}
  \Expect*{\FSBottom{testRStack}}{defaultValue}
  \FSShift{testRStack}
  \Expect*{\FSBottom{testRStack}}{defaultValue}
\end{qstest}
⟨/qstest⟩
```

Now we provide an example (non-linear) use of all of the `\FSPush`, `\FSPop`, `\FSUnshift`, and `\FSShift` macros

```
⟨*qstest⟩
\begin{qstest}{Full non-linear FIFO/stack integration test}
  {\FSCreate, \FSPush, \FSPop, \FSUnshift, \FSShift, \FSTop, \FSBottom}

  \FSCreate{testFS}{defaultValue}
  \Expect*{\FSTop{testFS}}{defaultValue}
  \Expect*{\FSBottom{testFS}}{defaultValue}
  \FSPush{testFS}{value1}
  \Expect*{\FSTop{testFS}}{value1}
  \Expect*{\FSBottom{testFS}}{value1}
  \FSUnshift{testFS}{value-1}
  \Expect*{\FSTop{testFS}}{value1}
  \Expect*{\FSBottom{testFS}}{value-1}
  \FSShift{testFS}
  \Expect*{\FSTop{testFS}}{value1}
  \Expect*{\FSBottom{testFS}}{value1}
  \FSPush{testFS}{value2}
```

```
  \Expect*{\FSTop{testFS}}{value2}
  \Expect*{\FSBottom{testFS}}{value1}
  \FSUnshift{testFS}{value-1again}
  \Expect*{\FSTop{testFS}}{value2}
  \Expect*{\FSBottom{testFS}}{value-1again}
  \FSUnshift{testFS}{value-2}
  \Expect*{\FSTop{testFS}}{value2}
  \Expect*{\FSBottom{testFS}}{value-2}
  \FSPop{testFS}
  \Expect*{\FSTop{testFS}}{value1}
  \Expect*{\FSBottom{testFS}}{value-2}
  \FSPop{testFS}
  \Expect*{\FSTop{testFS}}{value-1again}
  \Expect*{\FSBottom{testFS}}{value-2}
  \FSShift{testFS}
  \Expect*{\FSTop{testFS}}{value-1again}
  \Expect*{\FSBottom{testFS}}{value-1again}
  \FSPop{testFS}
  \Expect*{\FSTop{testFS}}{defaultValue}
  \Expect*{\FSBottom{testFS}}{defaultValue}
  \FSPop{testFS}
  \Expect*{\FSTop{testFS}}{defaultValue}
  \Expect*{\FSBottom{testFS}}{defaultValue}
  \FSShift{testFS}
  \Expect*{\FSTop{testFS}}{defaultValue}
  \Expect*{\FSBottom{testFS}}{defaultValue}
\end{qstest}
⟨/qstest⟩
```

Finally we add an integration test of the full suite, including `\FSClear`, `\FSSize`, and `\FSDestroy` macros

```
⟨*qstest⟩
\begin{qstest}{FSClear, FSSize, and FSDestroy test}
  {\FSCreate, \FSPush, \FSPop, \FSUnshift, \FSShift, \FSTop,
   \FSBottom, \FSClear, \FSDestroy, \FSSize}

  \FSCreate{testFSC}{defaultValue}
  \Expect*{\FSSize{testFSC}}{0}
  \Expect*{\FSTop{testFSC}}{defaultValue}
  \Expect*{\FSBottom{testFSC}}{defaultValue}
  \FSPush{testFSC}{value1}
  \Expect*{\FSSize{testFSC}}{1}
  \Expect*{\FSTop{testFSC}}{value1}
  \Expect*{\FSBottom{testFSC}}{value1}
  \FSUnshift{testFSC}{value-1}
  \Expect*{\FSSize{testFSC}}{2}
  \Expect*{\FSTop{testFSC}}{value1}
  \Expect*{\FSBottom{testFSC}}{value-1}
  \FSPush{testFSC}{value2}
  \Expect*{\FSSize{testFSC}}{3}
```

```
  \Expect*{\FSTop{testFSC}}{value2}
  \Expect*{\FSBottom{testFSC}}{value-1}
  \FSUnshift{testFSC}{value-2}
  \Expect*{\FSSize{testFSC}}{4}
  \Expect*{\FSTop{testFSC}}{value2}
  \Expect*{\FSBottom{testFSC}}{value-2}
  \FSClear{testFSC}
  \Expect*{\FSSize{testFSC}}{0}
  \Expect*{\FSTop{testFSC}}{defaultValue}
  \Expect*{\FSBottom{testFSC}}{defaultValue}
  \FSPush{testFSC}{value1}
  \Expect*{\FSSize{testFSC}}{1}
  \Expect*{\FSTop{testFSC}}{value1}
  \Expect*{\FSBottom{testFSC}}{value1}
  \FSUnshift{testFSC}{value-1}
  \Expect*{\FSSize{testFSC}}{2}
  \Expect*{\FSTop{testFSC}}{value1}
  \Expect*{\FSBottom{testFSC}}{value-1}
  \FSPush{testFSC}{value2}
  \Expect*{\FSSize{testFSC}}{3}
  \Expect*{\FSTop{testFSC}}{value2}
  \Expect*{\FSBottom{testFSC}}{value-1}
  \FSUnshift{testFSC}{value-2}
  \Expect*{\FSSize{testFSC}}{4}
  \Expect*{\FSTop{testFSC}}{value2}
  \Expect*{\FSBottom{testFSC}}{value-2}
  \FSDestroy{testFSC}
\end{qstest}
⟨/qstest⟩
```

3 Code for the FIFO/stack package

3.1 Identification

```
⟨*package⟩
\ProvidesPackage{fifo-stack}[2014/03/24 v1.0 Multi-FIFO/stack system]
\RequirePackage{ifthen}
```

3.2 FIFO/stack creation and destruction macro definitions

In the following code, we use only three counters (in the TEX meaning) we store the various values in macros, and use only the top, bottom or size counters when we need to do computations.

\tmp@fifo@stack@top

```
\newcount\tmp@fifo@stack@top
```

\tmp@fifo@stack@bottom

```
\newcount\tmp@fifo@stack@bottom
```

\tmp@fifo@stack@size

```
\newcount\tmp@fifo@stack@size
```

\fifo@stack@pointer All items in a given FIFO/stack named `fifoStackName` are stored in dynamically defined macros whose names consist of the `fifoStackName` followed by `@` followed by the value of the `\fifo@stack@pointer` macro with either `\tmp@fifo@stack@top` or `\tmp@fifo@stack@bottom` counters as argument.

We start by providing a unit test of the expected behaviour of the `\fifo@stack@pointer` macro.

```
⟨/package⟩
⟨*qstest⟩
\begin{qstest}{Unit test of \fifo@stack@pointer}{\fifo@stack@pointer}
  \makeatletter
  \Expect*{\fifo@stack@pointer{1}}{ii}
  \Expect*{\fifo@stack@pointer{0}}{i}
  \Expect*{\fifo@stack@pointer{-1}}{@i}
  \makeatother
\end{qstest}
⟨/qstest⟩
⟨*package⟩
```

Now the macrocode itself. Note the only two uses of the ε-TEX `\numexpr` macro.

```
\global\def\fifo@stack@pointer#1{%
  \ifnum#1<0
    @\expandafter\romannumeral\numexpr -1*#1 \relax
  \else
    \expandafter\romannumeral\numexpr #1+1 \relax
  \fi
}
```

\FSCreate Creating a new FIFO/stack is, essentially, creating new counter-like top, bottom and size macros, initializing them to 0, 1 and 0 respectively, as well as creating macros for stack-top and stack-bottom evaluation (see next two macros below).

```
%txs\FSCreate{fifoStackName}{defaultValue}#
\newcommand\FSCreate[2]{%
  \expandafter\gdef\csname #1@fifo@stack@count@top\endcsname{0}%
  \expandafter\gdef\csname #1@fifo@stack@count@bottom\endcsname{1}%
  \expandafter\gdef\csname #1@fifo@stack@count@size\endcsname{0}%
  \expandafter\do@fifo@stack@top@macro%
    \csname #1@fifo@stack@count@top\endcsname{#1}{#2}%
  \expandafter\do@fifo@stack@bottom@macro%
    \csname #1@fifo@stack@count@bottom\endcsname{#1}{#2}%
}
```

\do@fifo@stack@top@macro This stack-top evaluation macro is required due to an ugly trick. During the push, we "evaluate" the value to push with an `\edef`, because one might want to push something which contains the previous top of the stack (see the example package

provided). If the \FSTop or FSBottom macros compute the real value on the top or bottom of the stack, then, the expanded definition in \FSPush will contain more or less the *content* of \FSTop and not the value.

Another way would be to have a \csname...\endcsname pair within another one in an \ifx condition, which doesn't work.

```
\newcommand\do@fifo@stack@top@macro[3]{%
  \expandafter\newcommand\expandafter*%
  \csname fifo@stack@top@#2\endcsname{%
    \expandafter\ifx\csname #2@\fifo@stack@pointer#1\endcsname\relax
      #3%
    \else
      \csname #2@\fifo@stack@pointer#1\endcsname
    \fi
  }%
}
```

\do@fifo@stack@bottom@macro This is the stack-bottom macro defined for the same reasons as \do@fifo@stack@top@macro (see above).

```
\newcommand\do@fifo@stack@bottom@macro[3]{%
  \expandafter\newcommand\expandafter*%
  \csname fifo@stack@bottom@#2\endcsname{%
    \expandafter\ifx\csname #2@\fifo@stack@pointer#1\endcsname\relax
      #3%
    \else
      \csname #2@\fifo@stack@pointer#1\endcsname
    \fi
  }%
}
```

\FSClear While the size of the FIFO/stack is greater than zero, pop the FIFO/stack. Note the only use of the ifthen package's \whiledo macro.

```
%txs\FSClear{fifoStackName}#
\newcommand{\FSClear}[1]{%
  \whiledo{0 < \FSSize{#1}}{\FSPop{#1}}
}
```

\FSDestroy Clear the FIFO/stack and then undefine all of the supporting macros.

We provide a unit test of the expected behaviour of the \FSDestroy macro.

```
⟨/package⟩
⟨*qstest⟩
\begin{qstest}{Unit test of \FSdestroy}
  {\FSCreate, \FSDestroy}

  \FSCreate{testFSDestroy}{defaultValue}
  \makeatletter
  \ExpectIfThen{\isnamedefined{testFSDestroy@fifo@stack@count@top}}
  \ExpectIfThen{\isnamedefined{testFSDestroy@fifo@stack@count@bottom}}
  \ExpectIfThen{\isnamedefined{testFSDestroy@fifo@stack@count@size}}
```

```
  \ExpectIfThen{\isnamedefined{fifo@stack@top@testFSDestroy}}
  \ExpectIfThen{\isnamedefined{fifo@stack@bottom@testFSDestroy}}
  \makeatother
  \FSDestroy{testFSDestroy}
  \makeatletter
  \ExpectIfThen{\isundefined\testFSDestroy@fifo@stack@count@top}
  \ExpectIfThen{\isundefined\testFSDestroy@fifo@stack@count@bottom}
  \ExpectIfThen{\isundefined\testFSDestroy@fifo@stack@count@size}
  \ExpectIfThen{\isundefined\fifo@stack@top@testFSDestroy}
  \ExpectIfThen{\isundefined\fifo@stack@bottom@testFSDestroy}
  \makeatother
\end{qstest}
⟨/qstest⟩
⟨∗package⟩
```

And now the macrocode for `\FSDestroy`.

```
%txs\FSDestroy{fifoStackName}#
\newcommand{\FSDestroy}[1]{%
  \FSClear{#1}
  \expandafter\global\expandafter\let
    \csname #1@fifo@stack@count@top\endcsname=
      \fifo@stack@never@defined\relax
  \expandafter\global\expandafter\let
    \csname #1@fifo@stack@count@bottom\endcsname=
      \fifo@stack@never@defined\relax
  \expandafter\global\expandafter\let
    \csname #1@fifo@stack@count@size\endcsname=
      \fifo@stack@never@defined\relax
  \expandafter\global\expandafter\let
    \csname fifo@stack@top@#1\endcsname=
      \fifo@stack@never@defined\relax
  \expandafter\global\expandafter\let
    \csname fifo@stack@bottom@#1\endcsname=
      \fifo@stack@never@defined\relax
}
```

3.3 Stack based macro definitions

`\FSTop` I'd rather remove the "check" code, but, well, it would suppose the end-user of the macro is a fair developer, which is not realistic.

If the stack has been properly created, we call the top-stack evaluation macro.

```
%txs\FSTop{fifoStackName}#
\newcommand\FSTop[1]{%
  \ifx\csname #1@fifo@stack@count@top\endcsname\relax
    \PackageError{fifo-stack}{Undefined FIFO/stack #1}%
      {You should first create the FIFO/stack with \FSCreate}%
  \else
    \csname fifo@stack@top@#1\endcsname
  \fi
```

```
}
```

\FSPush The \FSPush macro is one of the most complex ones.

We provide a unit test of the ability to use \FSTop inside the value argument to a \FSPush macro (which is the reason for the complexity of this macro).

```
⟨/package⟩
⟨*qstest⟩
\begin{qstest}{Unit test of \FSPush and \FSTop}
  {\FSCreate, \FSPush, \FSTop}

  \FSCreate{testFSPush}{defaultValue}
  \Expect*{\FSTop{testFSPush}}{defaultValue}
  \FSPush{testFSPush}{\FSTop{testFSPush}-with-additional-text}
  \Expect*{\FSTop{testFSPush}}{defaultValue-with-additional-text}
\end{qstest}
⟨/qstest⟩
⟨*package⟩
```

And now the macrocode for \FSPush. First, we check if the stack is properly defined.

```
%txs\FSPush{fifoStackName}{value}#
\newcommand\FSPush[2]{%
  \ifx\csname #1@fifo@stack@count@top\endcsname\relax
    \PackageError{fifo-stack}{Undefined FIFO/stack #1}%
      {You should first create the FIFO/stack with \FSCreate}%
  \else
```

We store the new top-value in a macro, just because one might do

```
\FSPush{stack}{\FSTop{stack}.ext}
```

in which case the expansion of the value to push is something rather tricky.

Then, we step the top of the stack (put the top in the scratch counter, advance this counter, have the value back in the macro).

We separately adjust the size macro. We do this since actually computing the value in the \FSSize macro proved difficult.

Only after that, the value to be pushed is stored in the corresponding slot.

```
    \edef\fifo@stack@newtop{#2}%
    \tmp@fifo@stack@top\csname #1@fifo@stack@count@top\endcsname\relax
    \global\advance\tmp@fifo@stack@top by 1\relax
    \expandafter\xdef\csname #1@fifo@stack@count@top\endcsname
      {\the\tmp@fifo@stack@top}%
    \tmp@fifo@stack@size\csname #1@fifo@stack@count@size\endcsname\relax
    \global\advance\tmp@fifo@stack@size by 1\relax
    \expandafter\xdef\csname #1@fifo@stack@count@size\endcsname
      {\the\tmp@fifo@stack@size}%
    \expandafter\xdef\csname #1@\fifo@stack@pointer\tmp@fifo@stack@top\endcsname
      {\fifo@stack@newtop}%
  \fi
}
```

\FSPop The \FSPop macro is simpler: if the stack is properly defined and the stack-top is greater than or equal to the stack-bottom, then we undefine the existing 'top' macro and down-step the stack-top and stack-size.

```
%txs\FSPop{fifoStackName}#
\newcommand\FSPop[1]{%
  \ifx\csname #1@fifo@stack@count@top\endcsname\relax
    \PackageError{fifo-stack}{Undefined FIFO/stack #1}%
      {You should first create the FIFO/stack with \FSCreate}%
  \else
    \tmp@fifo@stack@top\csname #1@fifo@stack@count@top\endcsname\relax
    \tmp@fifo@stack@bottom\csname #1@fifo@stack@count@bottom\endcsname\relax
    \ifnum\tmp@fifo@stack@top<\tmp@fifo@stack@bottom\relax\else
      \expandafter\global\expandafter\let
        \csname #1@\fifo@stack@pointer\tmp@fifo@stack@top\endcsname=
          \fifo@stack@never@defined\relax
      \advance\tmp@fifo@stack@top by -1\relax
      \expandafter\xdef\csname #1@fifo@stack@count@top\endcsname
        {\the\tmp@fifo@stack@top}%
      \tmp@fifo@stack@size\csname #1@fifo@stack@count@size\endcsname\relax
      \global\advance\tmp@fifo@stack@size by -1\relax
      \expandafter\xdef\csname #1@fifo@stack@count@size\endcsname
        {\the\tmp@fifo@stack@size}%
    \fi
  \fi
}
```

Now, a debug only macro.

\FSShowTop

```
%txs\FSShowTop{fifoStackName}#
\newcommand\FSShowTop[1]{%
  {\edef\fifo@stack@top{\FSTop{#1}}%
   \typeout{The top of #1: \fifo@stack@top}}}
```

3.4 FIFO based macro definitions

\FSBottom

```
%txs\FSBottom{fifoStackName}#
\newcommand\FSBottom[1]{%
  \ifx\csname #1@fifo@stack@count@bottom\endcsname\relax
    \PackageError{fifo-stack}{Undefined FIFO/stack #1}%
      {You should first create the FIFO/stack with \FSCreate}%
  \else
    \csname fifo@stack@bottom@#1\endcsname
  \fi
}
```

\FSUnshift The \FSUnshift macro, like the corresponding \FSPush macro, is one of the most complex ones.

We provide a unit test of the ability to use `\FSBottom` inside the value argument to a `\FSUnshift` macro (which is the reason for the complexity of this macro).

```
⟨/package⟩
⟨*qstest⟩
\begin{qstest}{Unit test of \FSUnshift and \FSBottom}
  {\FSCreate, \FSUnshift, \FSBottom}

  \FSCreate{testFSUnshift}{defaultValue}
  \Expect*{\FSBottom{testFSUnshift}}{defaultValue}
  \FSUnshift{testFSUnshift}{\FSBottom{testFSUnshift}-with-additional-text}
  \Expect*{\FSBottom{testFSUnshift}}{defaultValue-with-additional-text}
\end{qstest}
⟨/qstest⟩
⟨*package⟩
```

And now the macrocode for `\FSUnshift`. First, we check if the stack is properly defined.

```
%txs\FSUnshift{fifoStackName}{value}#
\newcommand\FSUnshift[2]{%
  \ifx\csname #1@fifo@stack@count@bottom\endcsname\relax
    \PackageError{fifo-stack}{Undefined FIFO/stack #1}%
      {You should first create the FIFO/stack with \FSCreate}%
  \else
```

We store the new bottom-value in a macro, just because one might do

```
\FSUnshift{stack}{\FSBottom{stack}.ext}
```

in which case the expansion of the value to push is something rather tricky.

Then, we step the bottom of the stack (put the bottom in the scratch counter, decrement this counter, have the value back in the macro).

We separately adjust the size macro. We do this since actually computing the value in the `\FSSize` macro proved difficult.

Only after that, the value to be unshifted is stored in the corresponding slot.

```
    \edef\fifo@stack@newBottom{#2}%
    \tmp@fifo@stack@bottom\csname #1@fifo@stack@count@bottom\endcsname\relax
    \global\advance\tmp@fifo@stack@bottom by -1\relax
    \expandafter\xdef\csname #1@fifo@stack@count@bottom\endcsname
      {\the\tmp@fifo@stack@bottom}%
    \tmp@fifo@stack@size\csname #1@fifo@stack@count@size\endcsname\relax
    \global\advance\tmp@fifo@stack@size by 1\relax
    \expandafter\xdef\csname #1@fifo@stack@count@size\endcsname
      {\the\tmp@fifo@stack@size}%
    \expandafter\xdef\csname #1@\fifo@stack@pointer\tmp@fifo@stack@bottom\endcsname
      {\fifo@stack@newBottom}%
  \fi
}
```

\FSShift The \FSShift macro is fairly simple: if the stack is properly defined and the stack-top is greater than or equal to the stack-bottom, then we undefine the 'bottom' macro and up-step the stack-bottom and adjust the stack-size.

```
%txs\FSShift{fifoStackName}#
\newcommand\FSShift[1]{%
  \ifx\csname #1@fifo@stack@count@bottom\endcsname\relax
    \PackageError{fifo-stack}{Undefined FIFO/stack #1}%
      {You should first create the FIFO/stack with \FSCreate}%
  \else
    \tmp@fifo@stack@top\csname #1@fifo@stack@count@top\endcsname\relax
    \tmp@fifo@stack@bottom\csname #1@fifo@stack@count@bottom\endcsname\relax
    \ifnum\tmp@fifo@stack@top<\tmp@fifo@stack@bottom\relax\else
      \expandafter\global\expandafter\let
        \csname #1@\fifo@stack@pointer\tmp@fifo@stack@bottom\endcsname=
          \fifo@stack@never@defined\relax
      \advance\tmp@fifo@stack@bottom by 1\relax
      \expandafter\xdef\csname #1@fifo@stack@count@bottom\endcsname
        {\the\tmp@fifo@stack@bottom}%
      \tmp@fifo@stack@size\csname #1@fifo@stack@count@size\endcsname\relax
      \global\advance\tmp@fifo@stack@size by -1\relax
      \expandafter\xdef\csname #1@fifo@stack@count@size\endcsname
        {\the\tmp@fifo@stack@size}%
    \fi
  \fi
}
```

Again, a debug only macro for the FIFO usage.

\FSShowBottom

```
%txs\FSShowBottom{fifoStackName}#
\newcommand\FSShowBottom[1]{%
  {\edef\fifo@stack@bottom{\FSBottom{#1}}%
   \typeout{The bottom of #1: \fifo@stack@bottom}}}
```

3.5 FIFO/stack length macro definition

\FSSize We keep track of the changes to the stack-size in each of the \FSPush, \FSPop, \FSUnshift, and \FSShift macros, since actually computing the value in the \FSSize macro (below) was returning spurious tokens.

```
%txs\FSSize{fifoStackName}#
\newcommand\FSSize[1]{\csname #1@fifo@stack@count@size\endcsname}
```

3.6 Finishing off

```
⟨/package⟩
⟨*qstest⟩
\LogClose{lgout}
\stop
⟨/qstest⟩
```

4 Copyright and LaTeX Project Public License

```
txsBeginComment

Copyright (C) 2014 Stephen Gaito (PerceptiSys Ltd)

This work may be distributed and/or modified under the conditions of the
LaTeX Project Public License, either version  1.3  of  this license or (at
your option) any later version.

The latest version of this license is in
http://www.latex-project.org/lppl.txt and  version 1.3 or later is part of
all distributions of LaTeX version 2005/12/01 or later.

This work has the LPPL maintenance status ‘maintained’.

The Current Maintainer of this work is Stephen Gaito.

The released source can be found at:
https://github.com/diSimplex/latexFifoStack/tree/master/texStyle

The development source can be found at:
https://github.com/stephengaito/latexFifoStack/tree/master/texStyle

This work consists of the files fifo-stack.dtx, and fifo-stack.ins.

The command:
  pdflatex fifo-stack.ins
followed by:
  pdflatex fifo-stack.dtx
will produce the derived files: fifo-stack.sty, fifo-stack-test.tex,
and fifo-stack.pdf.

The command:
  pdflatex fifo-stack-test.tex
will regression test the fifo-stack package. Output will be found in
fifo-stack-test.lgout. If pdflatex completes with no errors, then all
regression tests passed.

txsEndComment
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Change History

1.0

General: Initial version 1