The enumerate package*

David Carlisle
2015/07/23

Abstract
This package gives the enumerate environment an optional argument which determines the style in which the counter is printed. An occurrence of one of the tokens A a I i or 1 produces the value of the counter printed with (respectively) \Alph \alph \Roman \roman or \arabic.

These letters may be surrounded by any strings involving any other \TeX expressions, however the tokens A a I i 1 must be inside a \{} group if they are not to be taken as special.

1 Examples

\begin{enumerate}
  \item one one one one one one one one one one
  \item two \begin{enumerate}
    \item one of two one of two
    \item two of two
  \end{enumerate}
\end{enumerate}

\begin{enumerate}
  \item one
  \item two\end{enumerate}

\begin{enumerate}
  \item one\begin{enumerate}[example a]
    \item one of two one of two
    \item two of two
  \end{enumerate}
\end{enumerate}

\begin{enumerate}
  \item one\begin{enumerate}[example b]
    \item one of two one of two
    \item two of two
  \end{enumerate}
\end{enumerate}

\begin{enumerate}
  \item one\begin{enumerate}[A-1]
    \item one
    \item two
  \end{enumerate}
\end{enumerate}

\label and \ref may be used as with the standard enumerate environment. \ref only produces the counter value, not the whole label. \ref prints the value

*This file has version number v3.00, last revised 2015/07/23.
in the same style as \item, as determined by the presence of one of the tokens A a i I i 1 in the optional argument. In the above example \ref{LA}, \ref{LB} and \ref{LC} produce ‘i’, ‘iiA’ and ‘1’ respectively.

2 Macros

1 (*package)
\begin{verbatim}
\enlab Internal token register used to build up the label command from the optional argument.
\newtoks\enLab
\enQmark This just expands to a ‘?’. \ref will produce this, if no counter is printed.
\end{verbatim}

3 \def\enQmark{?}

The next four macros build up the command that will print the item label. They each gobble one token or group from the optional argument, and add corresponding tokens to the register \enLab. They each end with a call to \enloop, which starts the processing of the next token.

\enLabel Add the counter to the label. #2 will be one of the ‘special’ tokens A a I i 1, and is thrown away. #1 will be a command like \Roman.
\begin{verbatim}
\def\enLabel#1#2{% \edef\enThe{\noexpand#1{\@enumctr}}\@enLab\expandafter{\the\enLab\csname the\@enumctr\endcsname}}
\end{verbatim}

\enSpace Add a space to the label. The tricky bit is to gobble the space token, as you can not do this with a macro argument.
\begin{verbatim}
\def\enSpace{\afterassignment\enSp@ce\let\@tempa= }\def\enSp@ce{\@enLab\expandafter{\the\enLab\space}\@enloop}
\end{verbatim}

\enGroup Add a \{ \} group to the label.
\begin{verbatim}
\def\enGroup#1{\@enLab\expandafter{\the\enLab{#1}}\@enloop}
\end{verbatim}

\enOther Add anything else to the label
\begin{verbatim}
\def\enOther#1{\@enLab\expandafter{\the\enLab#1}\@enloop}
\end{verbatim}

\enloop The body of the main loop. Eating tokens this way instead of using \@tfor lets you see spaces and all braces. \@tfor would treat a and \{a\} as special, but not \{a\}.
\begin{verbatim}
\def\enloop{\futurelet\en@temp\enloop@}
\end{verbatim}

\enloop@
\begin{verbatim}
\def\enloop@{% \ifx A\en@temp \def\en@tempa{\enLabel\Alph }\else \ifx a\en@temp \def\en@tempa{\enLabel\alph }\else \ifx i\en@temp \def\en@tempa{\enLabel\roman }\else \ifx I\en@temp \def\en@tempa{\enLabel\Roman }\else \ifx 1\en@temp \def\en@tempa{\enLabel\arabic}\else \ifx \@sptoken\en@temp \let\en@tempa=\enSpace \else \ifx \bgroup\en@temp \let\en@tempa=\enGroup \else \ifx \@enum@\en@temp \let\en@tempa=\enGobble \else \let\en@tempa=\enOther \fi\fi\fi\fi\fi\fi
\fi\fi\fi\fi\fi\let\en@tempa=\enloop@}
\end{verbatim}
Hook for possible extensions

\@enhook
\fi\fi\fi\fi\fi\fi\fi\fi

Process the current token, then look at the next.

\@tempa

\@enhook

Hook for possible extensions. Some packages may want to extend the number of special characters that are associated with counter representations. This feature was requested to enable Russian alphabetic counting, but here I give an example of a footnote symbol counter, triggered by *.

To enable a new counter type based on a letter, you just need to add a new \ifx clause by analogy with the code above. So for example to make * trigger footnote symbol counting, a package should do the following.

Initialise the hook, in case the package is loaded before enumerate.

\providecommand\@enhook{}

Add to the hook a new \ifx clause that associates * with the \fnsymbol counter command.

\g@addto@macro\@enhook{%
  \ifx *
    \edef\@tempa{\@enLabel\fnsymbol}%
    \fi
}

This code sequence should work whether it is loaded before or after this enumerate package. Any number of new counter types may be added in this way.

At this point we just need initialise the hook, taking care not to over write any definitions another package may already have added.

26 \providecommand\@enhook{}

\enumerate

The new enumerate environment. This is the first half of the original enumerate environment. If there is an optional argument, call \@enum@ to define the label commands, otherwise call \enumerate which is the second half of the original definition.

27 \def\enumerate{%
  \ifnum \@enumdepth >3 \@toodeep\else
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}\fi
  \edef\@enumctr{enum\romannumeral\the\@enumdepth}\fi
  \@ifnextchar[{\@enum@{}}
}

\@enum@

Handle the optional argument..

32 \def\@enum@[#1]{%
  Initialise the loop which will break apart the optional argument. The command to print the label is built up in \enlab. \enThe will be used to define \thenum.
  \enlab{}
  \let\enThe\enQmark

The \@enum below is never expanded, it is used to detect the end of the token list.

34 \@enloop\@enum
Issue a warning if we did not find one of the ‘special’ tokens.
\ifx\@enThe\@enQmark\@warning{The counter will not be printed.}\fi
Define \labelenum n and \thecenum n.
\expandafter\edef\csname label\@enumctr\endcsname{\the\@enLab}
\expandafter\let\csname the\@enumctr\endcsname\@enThe
Set the counter to 7 so that we get the width of ‘vii’ if roman numbering is in
force then set \leftmargin n. to the width of the label plus \labelsep.
\csname c@\@enumctr\endcsname7
\expandafter\settowidth
\csname leftmargin\romannumeral\@enumdepth\endcsname
{\the\@enLab\hspace{\labelsep}}%
Finally call \enumerate which is the second half of the original definition.
\enumerate
\enumerate All the list parameters have now been defined, so call \list. This is taken straight
from the original definition of \enumerate.
\def\@enum@{\list{\csname label\@enumctr\endcsname}}%
\usecounter{\@enumctr}\def\makelabel##1{\hss\llap{##1}}}
\enumerate