**macrolist** – Create lists of macros and manipulate them

Dennis Chen  
proofprogram@gmail.com  
v2.1.0, 2021/07/31*

---

**Abstract**

The **macrolist** package allows you to create lists and manipulate them, with utilities such as \macrolistforeach and an implementation of arr.join() from Javascript. Contrary to the name of the package, non-macros and groups of macros can be put into an item of the list.

---

**1 Usage**

The scope of lists is always global. This provides the most consistency and functionality for developers in places that are usually local (part of a group), such as environments and loops.

- **\macronewlist**

  To create a list, pass in \macronewlist{listname} to create a list with the name listname.

  The package checks that listname is not the name of another list, and will throw an error if another list listname has already been defined.

- **\macrolistexists**

  Writing \macrolistexists{listname}{true}{false} will execute true if listname exists and false otherwise.

- **\macrolistelement**

  To execute the ith element of listname, write \macrolistelement{listname}{i}.

  Note that lists are 1-indexed, meaning the first element is numbered 1, the second element numbered 2, and so on.

  An error will be thrown if listname is not a defined list, if i is empty, or if i is greater than the size of the list.

- **\macrolistindexof**

  2021/07/25 Fix behavior of listindexOf and listcontains for empty lists

  This works similar to indexof in almost any ordinary programming language. Write \macrolistindexof{list}{element} to get the index of where element first appears in list. If it never does, then the macro will expand to 0.

  The command uses \ifx instead of \if; this means that if you have \macro as an element with the definition this is a macro (assuming that this is a macro is

---

*https://github/com/chennisden/macrolist
not an element itself), then \macroindex{listname}{this is a macro} will expand to 0.

Because of the implementation of this macro, it can’t actually be parsed as a number. (See the ‘Limitations’ section for more information.)

\macrocontains

Writing \macrolistcontains{listname}{element}{true branch}{false branch} checks whether list listname contains element, executing true branch if it does and false branch if it does not.

\macrolistadd

To add something to the list listname, pass in \macrolistadd{listname}{position}{element}, where position is an optional argument. If nothing is passed in for position, then by default element will be added to the end of the list.

\macrolisteadd

To fully expand element before adding it to list listname, pass in \macrolisteadd{listname}{position}{element}.

This behaves similarly to \edef.

\macrolistremove

To remove an element in a list, write \macrolistremove{listname}{index}.

\macrolistremovelast

To remove the last element in a list, write \macrolistremovelast{listname}.

This behaves like C++’s pop_back.

\macrolistclear

To clear a list, write \macrolistclear{listname}.

\macrolistsize

To get the size of a list, write \macrolistsize{listname}.

\macrolistforeach

To write a for each loop, write

\macrolistforeach{listname}{\element}{begin}{end}{action}

Note that begin and end are optional arguments, and by default, they take the values 1 and \macrolistsize{listname}. If you pass in begin, you must also pass in end.

\macrolistjoin

Executing \macrolistjoin{listname}{joiner} returns all of the elements separated by joiner. This behaves like Javascript’s arr.join().

2 Example

Here is the source code for a small document using macrolist.

\documentclass{article}
\usepackage{macrolist}

\begin{document}

\macronewlist{mylist}
\macrolistadd{mylist}{Some text}
% List: Some text
\newcommand\macro{This is a macro}
\macrolistadd{mylist}{\macro}
% List: Some text, \macro
\macrolistelement{mylist}{1}
% Prints out "Some text"
\macrolistadd{mylist}{1}{Element inserted into beginning}
% List: Element inserted into beginning, Some text, \macro

\macrolistremove{mylist}{1}
% List: Some text, \macro

\macrolistforeach{mylist}{\element}{We're printing out $\textbf{\element}$.}
% We're printing out $\textbf{Some text}$. We're printing out $\textbf{\macro}$.  

\macrolistjoin{mylist}{, }% Some text, \macro
\end{document}

3 Limitations

The \macrolistindexof macro cannot be parsed as a number. This is because we have to compare each element of the list to the passed in element and requires storing the index in a macro, which requires some unexpandable macros. (This is why we do not directly use \macrolistindexof when defining \macrolistcontains.)

4 Implementation details

All internal macros are namespaced to prevent package conflicts.

\macrolist@exists One internal macro we use is \macrolist@exists{listname}, which checks that listname exists. It throws an error otherwise.

\macrolist@inbounds We use \macrolist@inbounds{listname}{index} to check that first, listname is a defined list using \macrolist@exists, and second, that index is within bounds. It throws an error otherwise.
\ifnum\numexpr#2 \relax>\macrolistsize{#1}
\PackageError{macrolist}
{Index out of bounds}
{The number you have passed in to the second argument of your command is out of the bounds of list '1'.}
\fi

\begin{changeHistory}
\begin{enumerate}
\item[v1.0.0] General: Initial version
\item[v1.0.1] General: Add “scope is always global” to documentation
Fix date in initial version
Changes entry
Fix v. appearing in front of date in document title
Make a couple of defs and lets global to prevent scoping issues
\item[v1.0.2] General: Added comment markers to remove pars and fix spacing in listforeach
\item Print changelog in documentation
\item[v1.1.0] General: Add listexists
\item[v1.1.1] General: Fix foreach doc by removing incorrect begin
\item[v1.2.0] General: Add listindexof and listcontains
\item[v2.0.0] General: Add “macro” in front of each external command (to avoid conflicts with etoolbox)
\item[v2.1.0] General: Add macrolisteadd
\end{enumerate}
\end{changeHistory}