

macrolist – Create lists of macros and manipulate them

Dennis Chen
proofprogram@gmail.com

v1.1.1, 2021/07/23*

Abstract

The `macrolist` package allows you to create lists and manipulate them, with utilities such as `\listforeach` 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 seems to provide the most

`\newlist` To create a list, pass in `\newlist{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.

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

`\listelement` To execute the `i`th element of `listname`, write `\listelement{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.

`\listadd` To add something to the list `listname`, pass in `\listadd{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.

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

`\listremove`

*<https://github.com/chennisden/macrolist>

To remove the last element in a list, write `\listremove{listname}`. This behaves like C++'s `pop_back`.

`\listclear`

To clear a list, write `\listclear{listname}`.

`\listsize`

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

`\listforeach`

To write a for each loop, write

```
\listforeach{listname}{\element}[begin][end]{action}
```

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

`\listjoin`

Executing `\listjoin{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}

\newlist{mylist}
\listadd{mylist}{Some text}
% List: Some text

\newcommand\macro{This is a macro}

\listadd{mylist}{\macro}
% List: Some text, \macro

\listelement{mylist}{1}
% Prints out "Some text"

\listadd{mylist}[1]{Element inserted into beginning}
% List: Element inserted into beginning, Some text, \macro

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

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

```

\listjoin{mylist}{, }
% Some text, \macro

\end{document}

```

3 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.

```

1 \newcommand*\macrolist@exists[1]{%
2   \ifcsname c@macrolist@list@#1\endcsname
3   \else
4     \PackageError{macrolist}
5     {The first argument is not a defined list}
6     {Make sure you have defined the list before trying to operate on it.}
7   \fi
8 }

```

`\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.

```

9 \newcommand*\macrolist@inbounds[2]{%
10   \macrolist@exists{#1}%
11   %
12   \if\relax\detokenize{#2}
13     \PackageError{macrolist}
14     {No number has been passed into the second argument of your command
15     }{Pass in a number to the second argument of your command.}
16   \fi
17   %
18   \ifnum\numexpr#2 \relax>\listsize{#1}
19     \PackageError{macrolist}
20     {Index out of bounds}
21     {The number you have passed in to the second argument of your command\MessageBreak
22     is out of the bounds of list '#1'.}
23   \fi
24 }

```