

macrolist – Create lists of macros and manipulate them

Dennis Chen
proofprogram@gmail.com

v1.2.0, 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 provides the most consistency and functionality for developers in places that are usually local (part of a group), such as environments and loops.

`\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.

`\listindexof` This works similar to `indexof` in almost any ordinary programming language. Write `\listindexof{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`

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

is not an element itself), then `\listindexof{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.)

`\listcontains`

Writing `\listcontains{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.

`\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}`.

`\listremoveat`

To remove the last element in a list, write `\listremoveat{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 Limitations

The `\listindexof` 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 `\listindexof` when defining `\listcontains`.)

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.

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 }
```

Change History

v1.0.0		to remove pars and fix spacing
General: Initial version 1	in listforeach 1
v1.0.1		Print changelog in
General: Add “scope is always		documentation 1
global” to documentation 1	v1.1.0
Fix date in initial version		General: Add listexists 1
changes entry 1	v1.1.1
Fix v. appearing in front of date		General: Fix foreach doc by
in document title 1	removing incorrect begin 2
Make a couple of defs and lets		v1.2.0
global to prevent scoping issues	1	General: Add listindexof and
v1.0.2		listcontains 1
General: Added comment markers		