The `modulus` Package, v1.0

Donald P. Goodman III

May 2, 2018

Abstract

TEX (traditional TEX, not even ε-TEX) has the four basic functions with `\advance`, `\multiply`, and `\divide`, though these are all integer operations. This tiny package makes a modulus operator usable with TEX (or LATEX, and probably most flavors). It also provides a non-destructive way of getting at the integer quotient of two numbers.

Contents

1 The Operators

One question immediately occurs to me: why bother with such a package, when ε-TEX already has `\numexpr`, and there are so many advanced packages providing real math, and even floating-point operations and bignums? The answer is twofold: (1) an interesting project, and (2) TEX is the archival format, as Knuth has always emphasized, and so doing things in ordinary TEX, where possible, should be preferred, to keep it such.

`\modulo` The core of the package is `\modulo`:

`\modulo {`⟨*dividend*⟩`} {`⟨*divisor*⟩`}`

`\remainder` Running this will assign the remainder of the integer division of *dividend* and *divisor* to the counter `\remainder`. The values of *dividend* and *divisor* will not be changed, if they are changeable (e.g., if they are counters).

Note that the package works with plain TEX counters, *not* LATEX-style counters created and set with `\newcounter{counter}` and friends. If you try to use it with `\thecounter` or `\value{counter}` in the LATEX style, it will not work. Fortunately, plain-TEX counters are perfectly usable within LATEX.

We can use the macro as such:

```
\modulo{\the\dividend}{\the\divisor}\the\remainder
```

Dividend	Divisor	Modulus	Dividend	Divisor
1	4	1	1	4
2	4	2	2	4
3	4	3	3	4
4	4	0	4	4
5	4	1	5	4
6	4	2	6	4
7	4	3	7	4
8	4	0	8	4
9	4	1	9	4

Table 1: A few example modulos, dividend and divisor both before and after the operation.

This will execute the macro and print the resulting remainder. We can also use simple number tokens in the same way:

`\modulo{8}{3}\the\remainder`: 2

In the table above, the divisor is always the value of a counter `\testdivisor`, which is 4; the dividend is the value of a counter `\testdividend`, which is reset as indicated in each line. We then do the modulus operation and print the value of `\remainder` in the "Modulus" column. We then reprint the values of `\testdividend` and `\testdivisor`, to show that we have not altered them by taking the remainder.

Plain TEX can also very simply take the integer quotient of a counter, with the `\divide` operator, like so:

```
\divide\thiscounter by4
```

However, this destroys the original value of `\thiscounter` (or whatever else one is dividing). The `modulus` package therefore also provides another macro, `\quotient` `\quotient`, which non-destructively takes the integer quotient of two numbers `\intquotient` and stores it in the counter `\intquotient`:

`\quotient{365}{4}\leavevmode\the\intquotient`: 91
`\modulo{365}{4}\leavevmode\the\remainder`: 1

Note that if `\the\thecounter` doesn't print as expected, it may be because you're still in vertical mode (the `\the` operator only works in horizontal mode), so try `\leavevmode` before printing the counter.

2 Implementation

Not much to this one. We define our counter to hold the eventual modulus:

```
1 \newcount\remainder%
```

Then we define the macro which will assign the modulus to that counter:

```
\def\modulo#1#2{%
\remainder=#1%
\divide\remainder by#2%
\multiply\remainder by#2%
\multiply\remainder by-1%
\advance\remainder by#1\relax%
}%
```

Then, for good measure, we also have a macro which will non-destructively take the integer quotient of a division, and store it in the counter `\intquotient`:

```
\newcount\intquotient%
\def\quotient#1#2{%
\intquotient=#1%
\divide\intquotient by#2%
}%
```

Happy TEXing!