
 前 30 个素数为 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, and 113。你可能觉得这没什么; 但是你可能吃惊的是上面这句话由下面所排版出:

前~30~个素数为~\primes{30}。

TeX 通过展开宏 `\primes` 执行了所有的计算, 因此作者可以确信上面所给素数列没什么输入错误。下面就是此宏集:


```
\newif\ifprime \newif\ifunknown % boolean variables
\newcount\n \newcount\p \newcount\d \newcount\a % integer variables
\def\primes#1{2,~3% assume that #1 is at least 3
  \n=#1 \advance\n by-2 % n more to go
  \p=5 % odd primes starting with p
  \loop\ifnum\n>0 \printifprime\advance\p by2 \repeat}
\def\printp{, % we will invoke \printp if p is prime
  \ifnum\n=1 and~\fi % 'and' precedes the last value
  \number\p \advance\n by -1 }
\def\printifprime{\testprimality \ifprime\printp\fi}
\def\testprimality{\d=3 \global\primetrue
  \loop\trialdivision \ifunknown\advance\d by2 \repeat}}
\def\trialdivision{\a=\p \divide\a by\d
  \ifnum\a>\d \unknowntrue\else\unknownfalse\fi
  \multiply\a by\d
  \ifnum\a=\p \global\primefalse\unknownfalse\fi}
```

这种计算相当直接, 除了它有一个循环在另一个循环中; 因此, `\testprimality` 引入了一个额外的大括号组, 来保证内部循环控制不受外部循环的影响。当 `\ifprime` 被设定为真或假时, 大括号要求编写使用 `'\global'`。TeX 处理此句所花费的时间比处理一个页面还要多; 宏 `\trialdivision` 被展开了 132 次。

 做这些漂亮工作的宏 `\loop` 实际上非常简单。它把假定要重复的代码放在叫做 `\body` 的控制系列中, 并且接着用另一个控制系列重复到条件为假:

```
\def\loop#1\repeat{\def\body{#1}\iterate}
\def\iterate{\body\let\next=\iterate\else\let\next=\relax\fi\next}
```

`\iterate` 的展开的结尾是 `\next` 的展开; 因此 TeX 在调用 `\next` 之前能把 `\iterate` 从内存中清除掉, 这样在长循环中内存就不会被塞满。计算机科学家称其为“末端回归”。

 下面的宏 `\hex` 把计数寄存器 `\n` 转换为十六进制表示, 它举例说明了一个递归控制结构, 许多 `\hex` 的副本都可以在其中同时使用。在这个应用程序中, 递归比简单的 `\loop` 重复更好, 因为十

六进制数字是从右到左算出来的, 而它们在输出时是从左到右。(在 `\n` 中的数字必须  $\geq 0$ 。)

```
\def\hex{\count0=\n \divide\n by16
  \ifnum\n>0 \hex\fi \count2=\n \multiply\count2 by-16
  \advance\count0 by\count2 \hexdigit}}
\def\hexdigit{\ifnum\count0<10 \number\count0
  \else\advance\count0 by-10 \advance\count0 by'A \char\count0 \fi}
```



我们的最后一个例子是计算变量中非空记号数目的宏; 例如, `\length{argument}` 展开为 `'8'`。它还举例说明了宏另一个方面的技术。

```
\def\length#1{\count0=0 \getlength#1\end \number\count0}}
\def\getlength#1{\ifx#1\end \let\next=\relax
  \else\advance\count0 by1 \let\next=\getlength\fi \next}
```