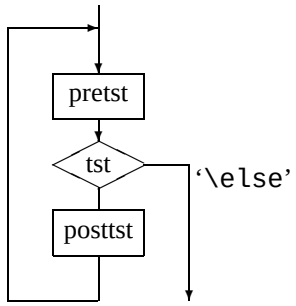
Paradigms: Loops

Kees van der Laan

1 BLUe's Design VII

Hi folks. When you like plain's `\loop`-s so much as I do then this is for you. Hang on Dek's loop implements the flow, The TEXbook 219

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



with as pseudosyntax for the tags

`\loop`⟨*pretst*⟩`\if...`⟨*posttst*⟩`\repeat`.

Special cases result when either ⟨*pretst*⟩, or ⟨*posttst*⟩ is empty. The former is equivalent to for example PASCAL's **while** ... **do** ..., and the latter to **repeat**... **until**. From this I conclude that all those `\while` casu quo `\repeat` flavors are not needed. Syntactic sugar? Yes, IMHO, with all respect.

In practice we all need repetitions either via a loop or via tail recursion, which by the way are equivalent. The loop notation is simpler to use than tail recursion, I guess.

2 Van der Goot's loop

Van der Goot implemented the loop construction as a straight tail recursion. An eye-opener I have simplified it into the following.[1]

```
\def\Loop#1\Pool{#1\Loop#1\Pool}
\def\Break#1\Pool{}
%a trivial example of use
\count0=10
\Loop a \advance\count0 by-1
   \ifnum\count0=0 \expandafter\Break\fi
\Pool
\end
```

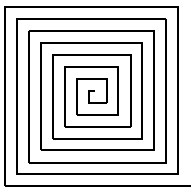
For more details see his Midnight suite, `loop.doc` and `loop.tex`.[2]

Remark. Either `\expandafter` is needed or the `\Break` should take `\fi` as replacement text. The above with `\expandafter` is convenient with nesting of loops.

3 Loops in markup

I used a loop in the markup for turtle graphics, especially for the stochastic Sierpiński carpets—to throw the dice repeatedly—and for a spiral. In the markup for the spiral below `\E`, ...`\N` mean draw in the directions east, ...north.

Example (Spiral)



```
$$\vbox to3cm{\vss
 \hsize0pt \offinterlineskip
 \unitlength.5ex \y0pt \x0pt
 \loop\N{\the\k}\advance\k+1
      \W{\the\k}\advance\k+1
      \S{\the\k}\advance\k+1
      \E{\the\k}\advance\k+1
 \ifnum\k<30 \repeat
\vss}$$
```

Remark. One can also start at a corner. However, for reuse of these kinds of pictures I consider it more convenient to start at the center of symmetry.

Most of the time I use the tail recursion via my FIFO or (binary) tree macros. My favorite FIFO example is the outline exercise The TEXbook ex11.5. I solved this via nested FIFO processing

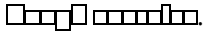
- words via `\fifow`, and
- characters via `\fifo`.

Example (Outlines)

```
\fifow Tough  exercise \wofif \unskip.
%with
\def\fifow#1 {\ifx\wofif#1\wofif\fi
   \processw{#1}\fifow}
\def\processw#1{\fifo#1 \ofif\ }
\def\process#1{\boxit#1}
\def\boxit#1{\setbox0=\hbox{#1}%
  \hbox{\lower\dp0\vbox{\hrule
   \hbox{\vrule\phantom#1\vrule}%
   \hrule}}}
```

[1] The unusual in the coding is the infinite loop. I have to unlearn much. In the old days of ALGOL 60 I was taught to avoid side-effects and infinite loops of course. With the expression language ALGOL 68 all storing was a side-effect. Now with the interpretive languages it is beneficial to think in infinite loops.

[2] Available on the TEX-Alive TUG CD.



Another nice example is writing lines verbatim to a file, that is manmac's `\copytoblankline` recast in FIFO terms.[3]

Example (Sorting citation lists)

Suppose, we have

```
\def\lst{\\\ia\\\ib\\\ic}
\def\ia{314}\def\ib{27}\def\ic{1}
```

then sorting, the invoke of `\lst`, yields 1, 27, 314.

The above is obtained as follows.

```
\def\dblbsl#1{\ifnum#1<\min\let\min=#1\fi}
%
\Loop\ifx\empty\lst\expandafter\break\fi
   \def\\{\let\\=\dblbsl\let\min=} %space
   \lst%find minimum
   \min%typeset minimum
   {\def\\#1{\ifx#1\min \else\nx\\%
    \nx#1\fi}\xdef\lst{\lst}}%
\Pool
%with auxiliaries
\def\Loop#1\Pool{#1\Loop#1\Pool}
\def\break#1\Pool{}
```

The coding implements the looping of the basic steps

- find minimum (via `\lst`, and suitable definition of Dek's active list separator `\\`)
- typeset minimum (via `\min`)
- delete minimum from the list (via another appropriate definition of the list element tag).

Remarks. `\dblbsl` is mnemonics for double backslash. The deletion of elements from the list works with `\ifx`. The variant with `\ifnum` does not work as such because a `\relax` is inserted by TEX.[4]

The problem occurred in sorting citation lists of bibliography references which are specified by their symbolic names.[5]

Example (Reading a file line by line)

```
\Loop\ifeof\readfile\Break\fi
  \read\readfile to\inputline
  <process \inputline>
\Pool
%with auxiliary
\def\Break#1\Pool{\fi}
```

This use occurred in the Convertor Assistant BLUe-2-LATEX, well ... it is the main loop.[6]

3.1 Relevancy

From my experience as exemplified above, I conclude that loops are mainly of interest for macro writers and not so much for the casual (La)TEX user.

4 Macro writers attention

Knuth's loop does not allow for the use of `\else` in the exit code because it is already part of the macro `\loop`. Kabelschacht 1987—and Spivak 1989—needed the use of `\else`.

Example (Kabelschacht's loop)

Kabelschacht removed the `\else` from the loop code as follows.

```
\def\loop#1\repeat{\def\iterate{#1\ea
   \iterate\fi}\iterate}
%a trivial example of use
\count0=10
\loop\advance\count0 by-1
   \ifnum\count0=0
   \else do what has to be done
\repeat
```

Remarks. The exit is via the then-branche in contrast with Knuth's loop. Suppressed are some efficiency aspects with respect to storage. Kabelschacht's claim that his loop is a *generalization* of plain's loop must be seen in the light of not being restricted to quit a loop via the else-branch.

The reason, I can think of, for introducing another loop macro, while the most general form has been implemented already, is the existence of commands like `\ifvoid`, and `\ifeof`, and the absence of their negatives `\ifnonvoid`, respectively `\ifnoneof`. In those cases we like to continue the loop via the `\else` branch. For the latter case this means to continue the loop when the file is *not* ended. This can be obtained via modifying the loop à la Kabelschacht or more elegantly via van der Goot's macro.

Another approach is to use a `\newif` parameter, better known as 'boolean' or 'logical' in other programming languages, together with Knuth's loop. A `\newif` parameter, `\ifneof`, can be used to test for an end of file or casu quo, an end of a list.

```
\ifx\lst\empty\neofalse\else\noetrue\fi\ifneo
```

Example (Reading a list via Knuth's loop)

```
\newif\ifneo
\def\lst{a b c}\neotrue
\loop\ifx\lst\empty\neofalse\fi
   \ifneo \def\lst{}
\repeat
```

Related to the coding of the logical ¬ are the codings of the logical `and`, ∧, and `or`, ∨, as can be seen from the accompanying table.

[3] See FIFO and LIFO sing the BLUes.

[4] The `\relax` can be suppressed by inserting an unexpandable token like `\noexpand \empty`. Courtesy Bernd Raichle.

[5] In BLUe's bibliography I have explained how to circumvent the sorting of citation lists of references.

[6] Line by line is a white lie. A line or a group embraced by curley braces is read.

Functional code	TEX coding
¬\if...	\if...\notfalse\else \nottrue\fi\ifnot
\if...∧\if...	\andtrue\if...\if... \else\andfalse \else\andfalse\fi\fi \ifand
\if...∨\if...	\ortrue \if...\else\if...\else \orfalse\fi\fi \ifor %alternative \orfalse \if...\ortrue\fi \if...\ortrue\fi \ifor

with the \newif-s: \ifnot, \ifand, and \ifor.

5 Nesting of loops

My favorite example of nesting of loops is the bubble sort.[7]

```
\def\bubblesort{%
%Data in defs \1, \2,...\<n>.
{\loop\ifnum1<\n{\k\n
  \loop\ifnum1<\k \advance\k-1
     \cmp{\deref\k}{\deref\n}%
   \ifnum1=\status\xch\k\n\fi
  \repeat}\advance\n-1
 \repeat}}%end \bubblesort
%with auxiliary
\def\deref#1{\csname\the#1\endcsname}
\let\cmp\cmpn %from blue.tex or provide
\def\cmp#1#2{
%Yields status=0, 1, 2 for =, >, <
...}
```

5.1 Inconsistency pitfall

I experience the non-expansion of a counter variable—or the non-dereference of it as you wish—within a \csname counter-intuitive. The counter must be preceded by \the, see \deref. I understand the difference with the situation that a value might be assigned but in this construct this is out of order.

Another nesting of 'loops' is in the earlier mentioned linear sorting of citation lists, where the 'inner loop' is the \xdef.

5.2 Braces around inner loops are mandatory

Pittman argued that there is a need for other loop codings.

> 'Recently, I encountered an application that required a set of nested loops and local-only assignments and definitions. TEX's \loop ... \repeat construction proved to be inadequate because of the requirement that the inner loop be grouped.'

In 'Syntactic Sugar' I have shown that his problem can be solved from a table point of view.[8]

However, Pittman was definitely right with respect to bracing the inner loop, because of the parameter separator \repeat. If braces are omitted, the first \repeat is mistaken for the outer one, with the result that the text of the outer loop will *not* become the first \body. The good way is, to make the inner \repeat invisible for the outer loop level, by enclosing the inner loop in braces.

With non-explicit nesting—for example the inner loop is the replacement text of a macro—we still need scope braces, because otherwise the \body of the outer loop will be silently redefined by the body of the inner loop.

5.3 Dialogue with TEX

Nesting of loops occurs when in dialogue with TEX only certain answers are allowed.

Example (Checking user input)

```
%<TeX_Marker>
%Insist on allowed answers only
\def\iw{\immediate\write0}
\def\readyesornotoanswer{{%
   \def\yes{yes}\def\no{no}
   \Loop\iw{Please provide yes or no:}
      \read-1 to\answer
      \ifx\answer\yes\ea\Break\fi
      \ifx\answer\no \ea\Break\fi
    \Pool\xdef\answer{\answer}}}
%
\let\yes\Break\let\no\relax
\endlinechar-1 %TB20.18
\Loop\message{Are you happy?}
   \readyesornotoanswer
   \ea\csname\answer\endcsname
   \iw{Too bad, I insist...}
\Pool
\bye
```

Remarks. The \xdef makes the answer globally available. En-passant a little 'switch functionality in TEX' is realized via \csname.

6 Hidden counter

I like a hidden counter very much.[9] In TEX we don't have a garbage collector and therefore there is no gains in storage. However, to alleviate the user from the details of the allocation of storage of the counter I provided the following.[10]

```
\def\preloop{%To create loopcnt, a
            %local loopcounter
   \bgroup \advance\count10 by 1
   \countdef\loopcnt=\count10
            %Symbolic name
   \loopcnt=1 %(default)
}%end \preloop
\def\postloop{\loopcnt=0 %Restore
   \egroup}%end \postloop
```

[7] \cmp stands for comparison. \xch stands for exchange. For a pseudocode see Paradigms: Sorting.

[8] It is also in the tables chapter of PWT.

[9] As in ALGOL 68, METAFONT/MetaPost, or PostScript.

[10] Bernd Raichle has implemented local storage allocation macros. Although I can see the benefits of his robust approach especially together with push-the-button packages like LATEX, I refrained from it, because for my applications I know that the bounds—only 256 locations are available—are not severe. Moreover, intelligibility is hampered, and I like to add just a little to TEX, as little as possible.

I used the above in the coding of the Tower of Hanoi game. The paradigm in there is that it is used together with `\aftergroup`—shortcut `\ag`—to create *dynamically* a data structure, i.e. the tower of which the size is specified by the user.

```
%   \def\I{\disksep\i\disksep\ii
%        \disksep\iii...\disksep\‘n’}
%next to the defs for \i,\ii,...\‘n’.
\preloop\ag\def\ag\I\ag{%
  \loop
    \ea\xdef\csname\romannumeral\loopcnt
           \endcsname{\the\loopcnt}
       \ag\disksep\ea\ag\csname
         \romannumeral\loopcnt\endcsname
    \ifnum\loopcnt<\n
    \advance\loopcnt by1
  \repeat   \ag}
\postloop
```

Note that `\disksep` is Dek's active list separator, which I use abundantly in macro writing. I call it the list element tag, because it is also needed before the first element.[11]

7 Flip-flop traversal

It occurs that loops are processed with the first or last traversal different from the others. In practice I needed—well, it was more elegant—in a play to handle only one player in each traversal but toggle the players: player, opponent, player, opponent, etc.

+	o	+
	o	+
o		+

The play at hand was tic-tac-toe, and the prototype implementation which demonstrates the toggling reads as follows.

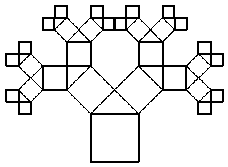
```
\def\play{\initialize
\loop\showboard
   \ifx\mark\markplayer\let\mark\markopponent
   \else\let\mark\markplayer
   \fi\iw{Supply index for \mark:}
      \read0to\index
   \ea\xdef\csname\index\endcsname{\mark}
\ifnum\index>0
\repeat}
\def\markplayer{+}\def\markopponent{o}
\endlinechar-1 %TB20.18
\play \bye
%with auxiliaries
%(\iw is shortcut for \immediate\write0)
\def\showboard{\iw{\1\2\3}...\iw{\7\8\9}}
\def\initialize{\def\1{-}...\def\9{-}}
```

In a general sense I need often in a repetitive situation to do something different at the beginning or at the end. Examples are my implementation of

- `\nitem`—numbered item and ilks
- to suppress the first headline of a chapter, and
- the `\\` in the linear sorting at the beginning.

8 Trees

Another intriguing repetitive or tail-recursive situation occurs when coding trees in TEX.[12] For fun—though the undertone is serious—I have added a tree from the Pythagorean family, borrowed from `pic.dat`.



The essence of the implementation in TEX of the repetition is shown below. For the full coding consult `pic.dat`.

```
\let\drawsq\ldrawsq %left drawing square
\def\pyth{\ifnum\level=1 \htyp\fi
    \drawsq\advance\level-1
    \multiply\kk18\divide\kk25
    {\turn7\x\leftx  \y\lefty
     \let\drawsq\ldrawsq\pyth}%
     \turn1\x\rightx \y\righty
     \let\drawsq\rdrawsq\pyth}
\def\htyp##1\pyth{\fi}
```

`\turn` turns over a multiple of 45°. (`\leftx`, `\lefty`) and (`\rightx`, `\righty`) contain the coordinates to start drawing a left square and a right square, respectively.

9 Conclusion

Many slightly different codings for a loop are around. This contributes to the reasons why understanding macro collections is so hard. It is so easy to come up with a variant, inhibiting intelligibility and trustworthiness.

I consider van der Goot's loop the simplest and most general, though Knuth's loop is usually sufficient for me.

10 Looking back

It is fun to flashback at for example Child's TEX Selftest, 1989, and to conclude that much of the perceived important TEXing nitty-gritties is not needed—not to mention all those essential issues which are lacking—when developing something like BLUe's format system.

This is precisely the reason why I pay so much attention to flashbacks, to summarize the paradigms, to bring to light what is needed in practical work, in such a way that the essence can be reused easily. Towards a discipline of TEXing. My case rest.

Have fun, and all the best

[11] I use them also with more than one argument, especially in selective loading of entries from a database. The above construction of a list is also used in my test example for sorting random words. Each character is obtained randomly and placed after the loop to form words of random length. Neat.

[12] Well also in METAFONT and PostScript.