1 Introduction

The \texttt{xfor} package redefines \texttt{\textbackslash for} so that the loop can be prematurely terminated, akin to \texttt{C}/Java’s \texttt{break} statement except that the loop will terminate at the end of the current iteration. The syntax for \texttt{\textbackslash for} remains the same:

\begin{verbatim}
\texttt{\textbackslash for}\langle cmd\rangle:=\langle list\rangle\textbackslash do\langle body\rangle
\end{verbatim}

where \langle cmd\rangle is a command name that is assigned to the current element of the list given by \langle list\rangle at each iteration.

To terminate the loop at the end of the current iteration, use the command \texttt{\textbackslash endfortrue}. This command may be used anywhere in \langle body\rangle, but will only take effect at the end of the current iteration. The remainder of the list is stored in \texttt{\textbackslash forremainder}. You can test whether the loop was prematurely terminated...
As from version 1.02, there is also provision for looking ahead. At each iteration in the loop, the next element is stored in `\@xfor@nextelement`. On the last iteration, this value will be `\@nil`, and so can be checked using

\[\ifx\@xfor@nextelement\@nnil\% last iteration\else\% not last iteration\fi\]

1.1 Example (ordered insertion)

Suppose you have list of sorted numbers stored in the command `\mylist`, e.g.:

\[
def\mylist{1,3,5,7,8,12,15,20}\]

and you want to insert a new value given by the command `\newval`, e.g.

\[
def\newval{11}\]

in the correct order. You can use `\@for` to iterate through each element in the sorted list, testing the value against the new value to be inserted. Once the new value has been inserted, the loop can be terminated, and any remaining elements can be appended to the new list. The following defines the command `\insertinto\{\newval\}\{\text{list}\}` which uses this method:

\[
\newcommand{\insertinto}[2]{%\\\def\nlst{}% new list initially empty\\\@for\n:=#2\do{%\\% store new list in \toks@
\expandafter\toks@\expandafter{\nlst}\\% test current value against new value\\\ifnum\n>#1\relax\\% new value needs to be inserted before current value\\\edef\newstuff{\number#1,\n}\\% end for loop at the end of this iteration\\\@endfortrue\\\else\\\edef\newstuff{\n}\\\fi\\% append new stuff to new list\\\ifx\nlst\@empty\\\edef\nlst{\newstuff}\\\else\\\edef\nlst{\the\toks@,\newstuff}\\\fi\\% check to see if for loop was prematurely terminated\\\if@endfor\\% loop may have been terminated during final iteration, in\\% which case \@forremainder is empty.\\\ifx\@forremainder\@empty\\% do nothing\\\else

2
% loop prematurely ended, append remainder of original list
% to new list
\expandafter\toks@\expandafter{\nlst}\
\edef\nlst{\the\toks@,\@forremainder}\
\fi
\else
% wasn’t prematurely terminated, so new value hasn’t been added
% so add now.
\expandafter\toks@\expandafter{\nlst}\
\ifx\nlst\@empty
\edef\nlst{\number#1}\
\else
\edef\nlst{\the\toks@,\number#1}\
\fi
\fi
\let#2=\nlst
\}

The \insertinto macro can then be used as follows:
\def\mylist{1,2,5,9,12,15,18,20}\
\def\newval{11}\
Original list: \mylist. New value: \newval.
\insertinto{\newval}{\mylist}
New list: \mylist.

1.2 Example (numerical insertion sort)

Care needs to be taken when nesting \@for-loops. Suppose you have a list of
unordered numbers, say
\def\mylist{4,2,7,1,10,11,20,15}
and you want to sort the list in numerical order using an insertion sort method.
To do this, a macro needs to be defined which iterates through each element in the
unordered list, and the element is then inserted into an ordered list. The previous
example described the macro \insertinto which does this, but this results in
nested \@for commands. The \insertinto command will need to be grouped to
avoid errors:
\newcommand*{\insertionsort}{#1}\
{\def\sortedlist{}\
@for\val:=#1\do{\insertinto{\val}{\sortedlist}}\
\let#1=\sortedlist}

This won’t work with the definition of \insertinto as given in the previous
section, as the grouping causes the definition of the sorted list to be localised to
that group. Replacing
\let#2=\nlst
with
\global\let#2=\nlst
at the end of the definition of \insertinto will fix that.
1.3 Example (looking ahead)

This example checks the next value to determine if the loop is on the last iteration, if it is, it does nothing, otherwise it does a semi-colon:

\makeatletter
\def\mylist{1,2,3,4,5}\
\@for\val:=\mylist\do{\val
  \ifx\@xfor@nextelement\@nnil \else ;\fi}
\makeatother

which produces: 1;2;3;4;5

2 Acknowledgements

Many thanks to Morten Høgholm for providing code to improve efficiency.

3 The Code

Note that the internal macros used by \@for have changed in version 1.04.

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{xfor}[2009/02/05 v1.05 (NLCT)]
\if@endfor
\Define a switch to determine if the for loop should be stopped prematurely.
\newif\if@endfor
\@gobbleseven
\Ignore seven arguments.
\long\def\@gobbleseven#1#2#3#4#5#6#7{}
\@for\@for\langle cmd \rangle := \langle list \rangle \do{\langle body \rangle}
\test if the list is empty and then re-arrange.
\long\def\@for#1:=#2\do{\%}
\Initialise
\@endforfalse
\def\@forremainder{}\%
\expandafter\def\expandafter\@fortmp\expandafter{#2}\%
\If list is empty do nothing.
\ifx\@fortmp\@empty
\expandafter\@gobbleseven\%
\fi
\expandafter\@@for\expandafter#1\expandafter{#2},\@nil,\@nil,\%
\@xfor@endmarker% magic end marker
\}
\@xfornoop
\Read up until magic end marker.
\long\def\@xfornoop#1\@xfor@endmarker{}
\@forloop \@forloop\{\var\}\{\langle\text{action}\rangle\}
\long\def\@forloop\#1\#2\#3,#4,\{\
\def\#1\{\#3\}\
\ifx\#1\@nnil
\long\def\@for@endmarker at the very end
\expandafter\@xforloopoop
\fi
\Removed \@xfor@storenext\#4,\@nil. Instead store next element in \@xfor@nextelement.
\def\@xfor@nextelement\{\#4\}\
\#2\%
\if@endfor
\expandafter\@xfor@ifgatherrest
\fi
\@xforloop\#1\{\#2\}\{\#4\},\%
\}

\@xifforrest Gather the remainder (and store in \@forremainder)
\long\def\@ifforrest\#1,\@nil,\@nil,\@xfor@endmarker\
\def\@fortmp\{\#3\}\
\ifx\@fortmp\@nnil
\def\@forremainder{}\
\else
\@forrest\{\#3\},\#4,\@xfor@endmarker\
\fi
\}

\@forrest Get remainder of list (stored in \@forremainder):
\long\def\@forrest\#1,\@nil,\@nil,\@xfor@endmarker\
\def\@forremainder\#1\
\}

4 Sample Document (sample.tex)

\listfiles
\documentclass{article}
\usepackage{xfor}
\makeatletter
% \insertinto{new value}{list}
\newtoks\tmptok
\newcommand{\insertinto}{\{\}
\def\nlist{}\%
\@for\n:=#2\do{}
% store new list in \tmptok
\expandafter\tmptok\expandafter{\nlist}\%
% test current value against new value
\ifnum\n\textgreater\#1\relax
\edef\newstuff{\number#1,\n}\
% end for loop at the end of this iteration
@endfortrue
\else
  \edef\newstuff{\n}\
  \fi
\fi
% append new stuff to new list
@ifx\nlst@empty
  \edef\nlst{\newstuff}%
\else
  \edef\nlst{\the\tmptok,\newstuff}%
\fi
% check to see if for loop was prematurely terminated
@if@endfor
  % loop may have been terminated during final iteration, in
  % which case @forremainder is empty.
 @ifx@forremainder@empty
    % do nothing
  \else
    % loop prematurely ended, append remainder of original list
    % to new list
    \expandafter\tmptok\expandafter{\nlst}%
    \edef\nlst{\the\tmptok,\@forremainder}%
  \fi
\else
  % wasn’t prematurely terminated, so new value hasn’t been added.
  % Add now.
  \expandafter\tmptok\expandafter{\nlst}%
  @ifx@nlst@empty
    \edef\nlst{\number#1}%
  \else
    \edef\nlst{\the\tmptok,\number#1}%
  \fi
  \fi
\global\let\lst2=\nlst
%
% \insertionsort{list}
% replaces list with sorted list
\newcommand{\insertionsort}[1]{%
  \def\sortedlist{}%
  @for\val:=#1\do{%\insertinto{\val}{\sortedlist}}%
  \let\#1=\sortedlist
%
\begin{document}
Unsorted list:
\def\mylist{4,2,7,1,10,11,20,15}\mylist.

\insertionsort{\mylist}%
Sorted list: \mylist.
Iterate through the list (next element in parentheses):
\makeatletter
\@for:n:=\mylist\do{%
\n\@xfor@nextelement\@nnil
% last iteration
\else
\@xfor@nextelement\);\fi
};
\end{document}

Change History

1.0
General: Initial version ........ 1
1.01
\@forgatherrest: made long .... 5
\@iforgatherrest: made long ... 5
1.02
General: Added \@xfor@storenext 5
1.04
\@for: Added by Morten Høgholm .................. 4
\@forgatherrest: argument syntax changed (MH) ............ 5
\@gobbleseven: added by Morten Høgholm ................. 4
\@xfornoop: \@forloop replaced by \@xfornoop to prevent conflict with \@tfor ................. 4
second and third arguments dropped (MH) ............ 4
General: removed \@iforloop ... 5
Removed \@xfor@storenext ... 5
1.05
\@forloop: \@forloop replaced by \@forloop to prevent conflict with other packages that use \@forloop ................. 5
Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in roman refer to the pages where the entry is used.

Symbols
\@for .................. 4 \@forgatherrest ............ 5 \@tfor ............. \@forloop ................. 1 \@forremainder ...... 1 \@gobbleseven ........ 4 \@iforloop .......... \@iforgatherrest .... 5 \@endfortrue ........ 1 \@xfor@nextelement .... 2 \@if@endfor ........ 2, 4