1 Getting started

1.1 A minimal file

Before using the listings package, you should be familiar with the LaTeX typesetting system. You need not to be an expert. Here is a minimal file for listings.

```
   \documentclass{article}
   \usepackage{listings}
%
   \begin{document}
   \lstset{language=Pascal}
%
     % Insert Pascal examples here.
%
   \end{document}
```

Now type in this first example and run it through LaTeX.

Must I do that really? Yes and no. Some books about programming say this is good. What a mistake! Typing takes time—which is wasted if the code is clear to you. And if you need that time to understand what is going on, the author of the book should reconsider the concept of presenting the crucial things—you might want to say that about this guide even—or you're simply inexperienced with programming. If only the latter case applies, you should spend more time on reading (good) books about programming, (good) documentations, and (good) source code from other people. Of course you should also make your own experiments. You will learn a lot. However, running the example through LaTeX shows whether the listings package is installed correctly.

The example doesn't work. Are the two packages listings and keyval installed on your system? Consult the administration tool of your TeX distribution, your system administrator, the local TeX and LaTeX guides, a TeX FAQ, and section **??**—in that order. If you've checked *all* these sources and are still helpless, you might want to write a post to a TeX newsgroup like `comp.text.tex`.

Should I read the software license before using the package? Yes, but read this *Getting started* section first to decide whether you are willing to use the package. ;-)

1.2 Typesetting listings

Three types of source codes are supported: code snippets, code segments, and listings of stand alone files. Snippets are placed inside paragraphs and the others as separate paragraphs—the difference is the same as between text style and display style formulas.

No matter what kind of source you have, if a listing contains national characters like é, Ł, ä, or whatever, you must tell the package about it! Section uSpecialCharacters discusses this issue.

Code snippets The well-known LaTeX command — `typesets code snippets verbatim.`The new command — pretty−prints the code, for examplethe code and can be replaced by any character not in the code; —var i : integer ;— gives the same result.

Displayed code The `lstlisting` environment typesets the enclosed source code. Like most examples, the following one shows verbatim LaTeX code on the right and the result on the left. You might take the right-hand side, put it into the minimal file, and run it through LaTeX. [lstlisting]

```
    for i:=maxint to 0 do
    begin
        { do nothing }
    end;
%
    Write('Case insensitive ');
    WritE('Pascal keywords.');
```

It can't be easier.

That's not true. The name '`listing`' is shorter. Indeed. But other packages already define environments with that name. To be compatible with such packages, all commands and environments of the listings package use the prefix '`lst`'. The environment provides an optional argument. It tells the package to perform special tasks, for example, to print only the lines 2–5:

```
    begin
        { do nothing }
    end;
%
```

Hold on! Where comes the frame from and what is it good for? You can put frames around all listings except code snippets. You will learn how later. The frame shows that empty lines at the end of listings aren't printed. This is line 5 in the example.

Hey, you can't drop my empty lines! You can tell the package not to drop them: The key 'showlines' controls these empty lines and is described in section **??**. Warning: First read ahead on how to use keys in general.

I get obscure error messages when using 'firstline'. That shouldn't happen. Make a bug report as described in section uTroubleshooting.

Stand alone files Finally we come to —, the command used to pretty-print stand alone files. It has one optional and one file name argument. Note that you possibly need to specify the relative path to the file. Here now the result is printed below the verbatim code since both together don't fit the text width.

```
%%
%% This is file `listings.sty',
%% generated with the docstrip utility.
%%
```

The spacing is different in this example. Yes. The two previous examples have aligned columns, i.e. columns with identical numbers have the same horizontal position—this package makes small adjustments only. The columns in the example here are not aligned. This is explained in section **??** (keyword: full flexible column format). Now you know all pretty-printing commands and environments. It remains to learn the parameters which control the work of the listings package. This is, however, the main task. Here are some of them.

1.3 Figure out the appearance

Keywords are typeset bold, comments in italic shape, and spaces in strings appear as ␣. You don't like these settings? Look at this: [basicstyle,keywordstyle,identifierstyle,commentstyle,stringstyle,showstringspaces] [basicstyle,keywordstyle,identifierstyle,commentstyle,stringstyle,showstringspaces]

```
    for i:=maxint to 0 do
    begin
        { do nothing }
    end;
%
    Write('Case insensitive ');
    WritE('Pascal keywords.');
```

You've requested white coloured comments, but I can see the comment on the left side. There are a couple of possible reasons: (1) You've printed the documentation on nonwhite paper. (2) If you are viewing this documentation as a `.dvi`-file, your viewer seems to have problems with colour specials. Try to print the page on white paper. (3) If a printout on white paper shows the comment, the colour specials aren't suitable for your printer or printer driver. Recreate the documentation and try it again—and ensure that the color package is well-configured. The styles use two different kinds of commands. —`| and || both take no arguments but |_ does; it underlines the following argument. In general, the` *`very last`* `command may read exactly one argument, namely some material the package typesets. There's one exception. The last command of basicstyle` *`must not`* `read any tokens---or you will get deep in trouble.`

`'|basicstyle=|' looks fine, but comments look really bad with '|commentstyle=|' and empty basic style, say. Don't use different font sizes in a single listing.`

`But I really want it! No, you don't.`

Warning `You should be very careful with striking styles; the recent example is rather moderate---it can get horrible.` *`Always use decent highlighting.`* `Unfortunately it is difficult to give more recommendations since they depend on the type of document you're creating. Slides or other presentations often require more striking styles than books, for example. In the end, it's` *`you`* `who have to find the golden mean!`

1.4 Seduce to use

`You know all pretty-printing commands and some main parameters. Here now comes a small and incomplete overview of other features. The table of contents and the index also provide information.`

Line numbers `are available for all displayed listings, e.g. tiny numbers on the left, each second line, with 5pt distance to the listing: [numbers,numberstyle,stepnumber,numbersep]`

```
    for i:=maxint to 0 do
    begin
        { do nothing }
    end;
```

Listing 1: A floating example

```
    for i:=maxint to 0 do
    begin
        { do nothing }
    end;
%
    Write('Case insensitive ');
    WritE('Pascal keywords.');


%
    Write('Case insensitive ');
    WritE('Pascal keywords.');
```

I can't get rid of line numbers in subsequent listings. '|numbers=none|' turns them off.

Can I use these keys in the optional arguments? Of course. Note that optional arguments modify values for one particular listing only: you change the appearance, step or distance of line numbers for a single listing. The previous values are restored afterwards. The environment allows you to interrupt your listings: you can end a listing and continue it later with the correct line number even if there are other listings in between. Read section ?? for a thorough discussion.

Floating listings Displayed listings may float: Don't care about the parameter caption now. And if you put the example into the minimal file and run it through LaTeX, please don't wonder: you'll miss the horizontal rules since they are described elsewhere.

LaTeX's float mechanism allows one to determine the placement of floats. How can I do that with these? You can write '|float=tp|', for example.