Pseudocode Listings in LyX

Paul A. Rubin (rubin@msu.edu)

January 12, 2022

1 Introduction

The **Pseudocode listing** module (file `pseudolst.module`) is intended to facilitate creating pseudocode listings in LyX, with program listing insets. It requires the `listings` LaTeX package. (Program listings can also use the `minted` package, but this module is incompatible with `minted`.)

As an example, here is a pseudocode listing for Euclid's algorithm to compute the greatest common divisor of two integers. To see how it looks, just compile this document (after installing the module).

```
1  Input: Positive integers m and n
2  Output: Greatest common divisor of m and n
3  function gcd(m, n) {
4     x ← max(m, n)
5     y ← min(m, n)
6     while (y > 0) {
7        z ← x mod y
8        x ← y
9        y ← z
10    }
11    return x
12 }
```

2 Installation

The first step is to make sure that you have the `listings` package. If you are not sure, on most systems you can open a command prompt / terminal and run '`kpsewhich listings.sty`'. That should return the location of the `listings` style file. If you get no output, `listings` is not installed, and you will need to install it the same way you install other LaTeX packages.

The second step is to install the `pseudolst.module` file in your local layouts directory. If you are not sure where that is, select `Help > About LyX`, find the entry marked "User directory:", and look for a folder named layouts under that directory. Plop the module file into that folder.

The third and final step is to reconfigure LyX and restart it.

3 Usage

To insert a pseudocode listing, first create a program listing inset via `Insert > Program Listing`. The `Embedded Objects` help file contains details about usage of program listing insets, which I will not repeat here. There are a few tricky bits that I will explain.

After inserting an empty program listing inset, right click it and select `Settings...`, then make any adjustments you want. On the "Main Settings" tab, leave the language in its default setting ("No language").

On the "Advanced" tab, I recommend you make any parameter changes *before* setting the language. Once you are ready, add "language=pseudocode" (without the quotation marks) in the right-hand box and select the option "Bypass validation". (The reason for making other settings changes first is that LyX offers a nice feature that lets you type a question mark (?) in the right-hand box and get help with parameter settings. Selecting "Bypass validation" turns that feature off. Not selecting "Bypass validation" after setting the language will cause LyX to obsess about the fact that pseudocode is not one of the predefined languages.)

Now just type your pseudocode into the inset. No special formatting will be done within LyX, but in the output recognized keywords will be highlighted, lines will be numbered (if you so choose in the settings), etc.

3.1 Multiple listings

If you plan to include multiple pseudocode listings in a document, you can make the aforementioned settings once, rather than repeating them for each listing. Use Document > Settings... > Listings to set document-wide defaults for all listings.

3.2 Line formatting

Neither this module nor the code listing inset dictates a particular layout (indentation style). You can use spaces or tabs to indent lines to your own taste. In the advanced settings of the listing inset, use "tab=<some number>" to set the number of spaces to which each tab character expands, and "lineskip=<dimension>" to add space between lines.

3.3 Mathematics

The module recognizes dollar signs ($) as math delimiters. This allows you to enter mathematical notation in your listings. LyX will not open a math inset inside a program listing, however, so you have to enter raw LaTeX between the dollar signs. As the embedded objects manual points out, you can still use LyX's math editing features, albeit indirectly. Open a math inset outside the listing and create your formula. Select the entire contents of the inset (but not the inset itself!), copy it to the clipboard, then paste it between the dollar signs in the listing.

If you need to include a literal dollar sign in your listing, you will have to embed it in a mathematical formula. Enter "$\$$" (without the quotes) to get a simple dollar sign in the output.

3.4 Raw LaTeX in listings

The module reserves the backtick (`) as an escape character. Anything between a pair of backticks is treated as raw LaTeX. If you need literal backticks in your listings, you can change the "escapechar={`}" line in the module (or delete it if you never need to enter raw LaTeX). Alternatively, you can just surround a backtick with dollar signs to treat it as a literal.

```
This uses raw LaTeX to typeset "LaTeX": LaTeX
```

3.5 Comments

The module is configured to treat "//" as the start of an inline comment and "/* ... */" as a longer, or multi-line, comment. The following example includes a few comments.

```
1  procedure erase(list) {
2    /* Scan the list and delete any naughty words. */
3    for (item in list) {
4      if (isNaughty(item)    // isNaughty() does what it sounds like
5      then remove(item)
6    }
7  }
```

As with everything else, you can edit the module file to change the comment delimiters if you wish.

3.6 Plain text in listings

If you need to plain text into a listing, consider surrounding it with backticks. This will alter the formatting to what is likely something more visually pleasing, as the following example shows.

```
This is text entered as-is.
```
This is text surrounded by backticks.

3.7 Vertical bars

Some people like to provide visual cues for the span of a loop or code block by providing a vertical line. The `listings` package does not support this, but the module provides a (somewhat tedious) kludge. It defines a command (`\vbar`) that draws a vertical line character with a little bit of space in front of it (to help align it with the character above it). If you type "`‘\vbar‘`" (without the double quotes) where you would like the vertical line, the output will (hopefully) (maybe) have vertical lines in the correct places. The following listing demonstrates this.

```
if (some condition)
| eat
| drink
| be merry
else
| pay your taxes
| drink even more
endif
```

If you want to adjust the horizontal spacing of the vertical bars, feel free to edit the module file. The default definition of `\vbar` is `\,\vrule`; try changing `\,` to something else (perhaps `\enspace`) to shift the line horizontally.

You may note that the keywords in this example are colored. Look at the advanced tab of the settings for the listing to see how.

3.8 Line references

It is common to refer to specific lines in a listing. This can be done using the pseudocode module, but it is a bit finicky compared to the way LyX normally handles cross-references.

You will want to turn on line numbering (in the settings dialog for your listing). Ordinarily, your next step would be to use the `Insert` menu to insert a label in the line you wish to reference, but unfortunately LyX disables label insertion inside listings. So we do this manually, by typing `‘\label{some label}‘` in the desired location.

Placing the cross-reference in the text can be done with `Insert > Cross-Reference...`, with a couple of caveats. First, the cross-reference dialog will not find your label, so you will have to type it into the "Selected

Label:" box. Second, the inserted cross-reference will be marked "BROKEN" by LyX, so you will need a little faith.

We repeat Euclid's algorithm below, this time containing a reference to the line with the modulus operator.

```
1  Input: Positive integers m and n
2  Output: Greatest common divisor of m and n
3  function gcd(m, n) {
4     x ← max(m, n)
5     y ← min(m, n)
6     while (y > 0) {
7        z ← x mod y
8        x ← y
9        y ← z
10    }
11    return x
12 }
```

The modulus operator occurs in line 7.

4 Keywords

The following keywords are recognized by the module:

- begin
- break
- do
- else
- else if
- end
- endfor
- endif
- endwhile
- for
- foreach
- function
- if
- in
- Input:
- Output:
- procedure
- repeat

- Require:
- return
- then
- until
- while

Keywords are case-sensitive (a setting you can adjust in the module, by changing "sensitive=true" to "sensitive=false"). A few keywords contain a colon (:). Omission of the colon will cause the word not to be recognized as a keyword.

4.1 Additional keywords

One reason packages such as `listings` do not include pseudocode is that there is no defined standard for it. Everyone has their own style and preferences. If you want to include keywords not recognized by the module (or stop recognizing current keywords), you can just edit the `pseudolst.module` file in a text editor. Look for "morekeywords={...}" and tweak the list at will. Feel free to suggest new keywords using the issue tracker for repository where you found the module.

4.2 Escaping keywords

There may be times when you want a keyword not to be highlighted. Keywords are automatically not highlighted in comments. If you need a keyword to be ignored in another context, surrounded it with escape characters (backticks, unless you have changed that in the module file). The next example shows this.

```
begin // begin is highlighted at the start of the line but not in this comment
  wait for the dance to begin  // neither begin nor for is highlighted
  dance
end
```