Literate programming practice (C)

Table of Contents

1. README

2. This file is a practice file for using the GNU Emacs editor with

Org-mode as a C programming Integrated Development Environment (IDE).

3. You will learn how to:

- Create a sub-heading
- Create a task list
- Add meta header arguments
- Create a C code block
- Run a C program

4. Time: approx. 30-45 min.

5. When you're done with a section move the cursor on the section

heading and type `S-<right>` (or SHIFT+<right-arrow>).

6. `DONE` Create a sub-heading

Here you create a sub-heading and then customize it in various ways.

1. Create a sub-heading by entering `** Subheading` *in the first column*.
2. Below the sub-heading, enter `TAB` to auto-indent. Write a few words, then press `RET` (the "Enter" key) to get to the next line.

3. In the next line, type a sentence that's longer than 70 characters. When you're done, type `M-q` to auto-wrap the paragraph.
4. You can go up and down between headings with `C-x C-p` and `C-x C-n`.
5. You can restrict the buffer to a heading with `C-x n s` (and undo the restriction with `C-x n w`.
6. You can mark a heading as `TODO` or `DONE` when the cursor is on the heading and you enter `S-<left>` or `S-<right>`.
7. You can give the heading a priority `~[#A]` to `~[#C]` with `S-<up>` or `S-<down>`.

6.1. Subheading

Some sample text. Some lbbonger sample text. This line needs to be longer than 70 characters to demonstrate the auto-wrapping.

7. DONE Create a task list

Here you create a simple list, then continue it as a task list.

1. Below `-----` , enter `TAB` on a new line followed by `* milk`
2. At the end of this line, enter `M-RET`. This will create a new list item. Do this a few times and enter `honey`, `bread`, and `butter`
3. With the cursor anywhere in the list, enter `S-<left>` a few times to see the different bullet types
4. Move any list item up or down with `M-<up>` or `M-<down>`
5. Go to the end of the list (cursor after `butter`)
6. Enter `M-S-RET` to generate task items: `shop`, `sleep`, and `swim`.

7. Go with the cursor on any of the lines last created and type `C-c C-C` to toggle `[X]` and `[ ]`.

---

   1. milk
   2. honey
   3. bread
   4. butter
   5. [X] shop
   6. [X] sleep
   7. [ ] swim

8. DONE Establish meta header to run code blocks

Here you create a C source code block with header arguments.

1. Go to the top of the file (`M-<`)

2. Enter the following lines as a meta header:

```
#+TITLE: Emacs Org-mode practice file
#+AUTHOR: [yourName] (pledged)
#+PROPERTY: header-args:C :main yes :includes <stdio.h> :results output
```

3. Put the cursor on the line with `#+PROPERTY` and enter `C-c C-c`. You should see the message `Local setup has been refreshed` in the *echo area* (also called *mini buffer*) at the bottom of the screen.

9. DONE Create a code block

1. Type `TAB <s TAB` below (that is: TAB-key + < + s + TAB-key)
2. Type `C` on the header line (right where your cursor is). It should look like this: `#+begin_src C`
3. This is now a C source code block. Name the code block by adding `#+name: 1st_pgm` right above the `#+begin_src C`.

4. Add some C code between the `#+begin_src` and `#+end_src`. Click `TAB` to auto-indent lines or `M-q` to auto-indent a marked region. Enter the following two lines (or copy and paste them):

   puts("To C or not to C,"); puts("that is the question.");

   The result should show some *syntax highlighting* - the layout highlights structures of the programming language. Here is one example, and here is another one.

```
puts("To C or not to C,");
puts("that is the question.");
```

10. DONE Create another code block

1. Create another code block and name it `2nd_pgm`.

2. Add more header arguments after `#+begin_src C`. The header line should have the following arguments - each separated by one space:

   :main yes :includes <stdio.h> :results output :tangle pgm.c

3. Copy the C statements from the block 1.

```
puts("To C or not to C,");
puts("that is the question.");
```

11. Reference a code block using its `#+name`

You can use the header argument `:noweb yes` to tangle named code chunks into other code chunks. In the following chunk 1 this argument is set and the chunk 1 is inserted as `<<1st_pgm>>`.

```
puts("To C or not to C,");
puts("that is the question.");
puts("Another line");
```

The file tangles identical to the original file (except for the additional `puts` statement).

12. DONE Run the code blocks

1. To run each code block, put the cursor on any of its five lines and enter `C-c C-c` (or enter `M-x org-babel-execute-src-block`).
2. You should see the message `Code block evaluation complete.` in the minibuffer at the bottom, and the `#+RESULTS:` after each code block. Note that the results are named, too.

13. DONE Tangle and run a code block on the shell

1. Move the cursor anywhere in 1 and type `C-c C-v t` (or type `M-x org-babel-tangle`).
2. The mini-buffer should show the message: `Tangled 1 code block from practice.org`.
3. Type `M-x shell`. A terminal buffer opens below this file.
4. Go to the other buffer with `C-x o`.
5. Check that `pgm.c` is there with the command `ls -l`
6. Compile the file with `gcc pgm.c -o pgm`
7. Check that the executable program `pgm` is there
8. Run the executable with `./pgm`
9. Remove the other buffer with `C-x 0`
10. Save this file with `C-x C-s` and upload it to Schoology.

Author: [yourName] (pledged)

Created: 2022-05-21 Sat 15:08