

## Laboratory Exercises, Part 1

Deadline 2018-09-27

by Jens Eliasson, 2018-09-10

In this lab, you will continue to improve your knowledge in software development in C on Linux.

The goal of this lab is to teach you on how slightly larger programs are developed by combining the basic building blocks that exist in modern programming languages. You should also write a short technical report, containing information about how you solved the A1000 task. The report should be clear, and contain a short introduction to the task, how you solved the problem and if any specific problems occurred.

In [1], you will find the manuals of all available functions in the C standard library, search for example for *printf()*, *puts()*, *gets()*, *getchar()* and study how to use C functions. These functions are very important in this lab as you will make extensive use of them. The *printf()* function can for example be found using this direct link: <http://linux.die.net/man/3/printf> (for other functions replace printf with the desired function, e.g. fprintf, scanf, sscanf, ...)

This lab consists of two distinct parts, plus preparations for lab 2;

1. Mandatory assignments from the book. These should be completed and presented to a lab assistant before continuing with the next part,
2. Completing the program to calculate the dimensions of an Ax paper, where x is a standard paper, for example A0, A4, A10, and A1000. Write a two-page report describing your solution, any problems, testing, suggestions for improvements, etc.
3. Preparations for lab3. Here you will create a code skeleton for command input for lab3.

### Lab1 part 1 - Mandatory exercises

The following lists of exercises from the book [2], [4] must be completed and presented to a lab assistant before part two on this lab is started. **All exercises must be done individually or in groups of 2.** Each chapter of the course book, and thus lecture, has its own exercises. The exercise notion of x.y means Chapter x, Exercise y.

- Fourth edition:
  - Oral presentation of (tested) programs – 5.2, 5.3, 5.5, 6.2, 6.4, 7.2, 7.10, 7.11
- Third edition:
  - Oral presentation of (tested) programs – 6.2, 6.3, 6.5, 7.2, 7.4, 8.2, 8.10, 8.11

### Lab1 part 2 – Software development

In this part, you will complete a quite different task, namely performing a scientific calculation.

### Task 1

In this task, you will implement a program to answer one of life's fundamental questions: *What is the size of an A1000 paper?* The definition of the A-paper series is found in [3].

To complete this task, you must create a program that fulfills this task: *“What is the size of an Ax paper, if an A4 is 210 by 297 millimeters and x can range from 0 to 1000”*. See lecture 5 for more details.

Input to program: the size Ax to calculate, where x is a non-negative integer.

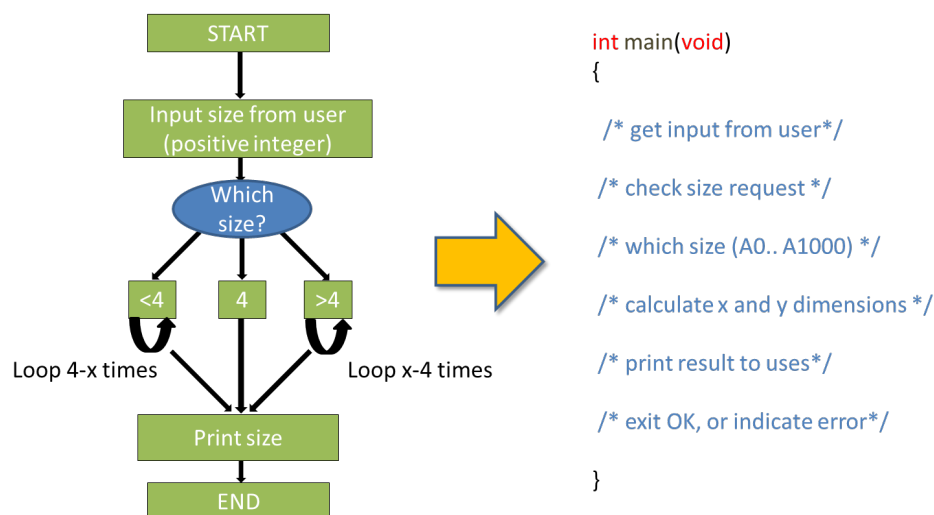
Calculations needed:

If x is 4, return 210x297 mm

If x is less than 4, double A4 4-x times (see [3])

If x is higher than 4, half A4 x-4 times (see [3])

Output of program: size, x and y dimensions.



You must use the proposed software design above, and create a code skeleton. After that, write course-grain code to match the comments. Finally, make the program compile by adding all the necessary variables, type definitions, brackets, etc.

When the program is completed and executed, it must first ask the user which A-size to calculate using the string **“Which size do you want to calculate?\nAx”** (where x is input by the user), and then perform the calculations. Finally, the program must print the exactly following string: **“A paper of size Ax is x by y mm”**. For example, in the case of an A5 the printout should be **“A paper of size A5 is 148 by 210 mm”**. If a user enters an incorrect size, the program must print a suitable error message and exit. The following are examples of incorrect input that must be detected by the program; , “A4.5”, “A6w”, “A4.0” or “A-2”, “A 3”

### Lab submission

You must do a Canvas submission of the A4 task in this lab plus all book assignments. You are required to show your lab assistant your results for all tasks upon request. Do not submit your files to Canvas unless a lab assistant has approved your solutions. Note that not all assignments are verified, but you are still expected to complete all of them found in the course manual. Also, add your report in PDF format to the zip file containing all your files. You must submit all your files inside a single .zip archive with the following name: yourusername\_lab1.zip, for example **jeneli-8\_lab1.zip**. Any other name of the file will be silently discarded and considered as no submission. If you do the lab in a group, it is enough with only one submission but both group members must present their solutions to a lab assistant.

### Lab1 part 3 – Preparations for lab 2

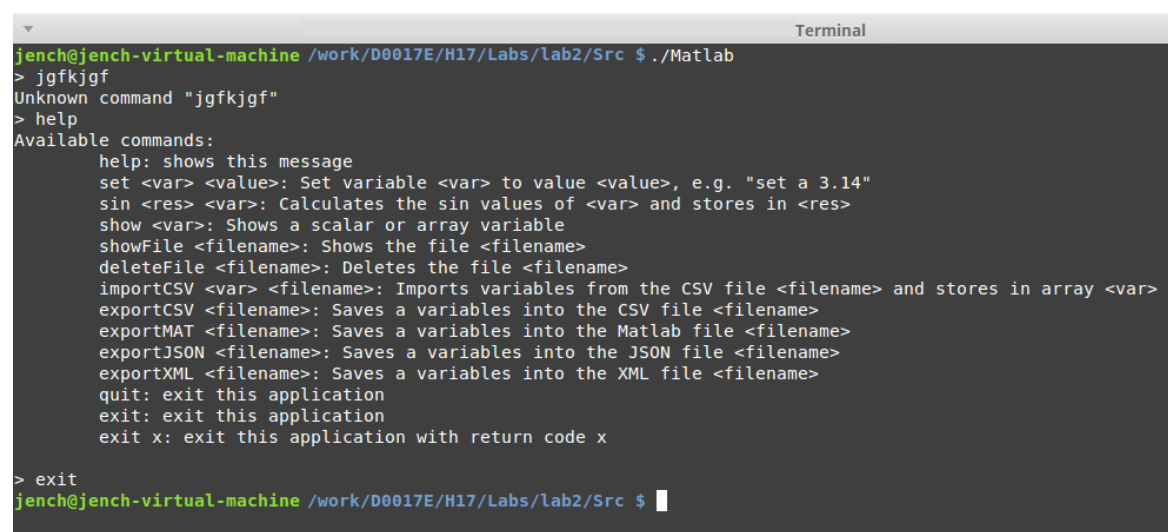
This part is a preparation for lab2, with a simple MATLAB [5] clone. Start by creating a new Code::Blocks C project and save everything in the folder **courses/D0017E/lab2**. Create the following files:

lab2.c lab2\_funcs.c lab2.h lab2\_funcs.h

Now, create the main function in lab2.c and make it receive the argc/argv arguments. If no arguments are given, i.e. argc is 1, then a loop should be used to capture all lines of text entered by the user. If the commands "exit" or "quit" are written, then the program must exit. If the command "help" is entered the program must print the help text below. When lab2 starts, you will implement support for all the other commands.

Use the following functions:

```
int main(int argc, char *argv[]) // (in lab2.c)
void printhelp(void); // prints the help message (in lab2_funcs.c)
int processLine(const char *line); // parses a line of text (in lab2.c)
```



```
jench@jench-virtual-machine /work/D0017E/H17/Labs/lab2/Src $ ./Matlab
> jgfkjgf
Unknown command "jgfkjgf"
> help
Available commands:
  help: shows this message
  set <var> <value>: Set variable <var> to value <value>, e.g. "set a 3.14"
  sin <res> <var>: Calculates the sin values of <var> and stores in <res>
  show <var>: Shows a scalar or array variable
  showFile <filename>: Shows the file <filename>
  deleteFile <filename>: Deletes the file <filename>
  importCSV <var> <filename>: Imports variables from the CSV file <filename> and stores in array <var>
  exportCSV <filename>: Saves a variables into the CSV file <filename>
  exportMAT <filename>: Saves a variables into the Matlab file <filename>
  exportJSON <filename>: Saves a variables into the JSON file <filename>
  exportXML <filename>: Saves a variables into the XML file <filename>
  quit: exit this application
  exit: exit this application
  exit x: exit this application with return code x
> exit
jench@jench-virtual-machine /work/D0017E/H17/Labs/lab2/Src $
```

Figure 1: Matlab-clone test

## References

1. C functions manual, <http://linux.die.net/man/3/printf>
2. Programming in C, third edition, Stephen G. Kochan
3. A paper sizes, quick lookup. <http://www.papersizes.org/a-paper-sizes.htm>
4. Programming in C, fourth edition, Stephen G. Kochan
5. MathWorks MATLAB, <https://se.mathworks.com/products/matlab.html>