Systems Software

VISUALIZATIONANDMULTIMEDIALAB

BINF31aa (BINF3101)

Prof. Dr. Renato Pajarola

Exercise Completion Requirements

- Exercises are mandatory, at least 3 of them must be completed successfully to finish the class and take part in the final exam. The first exercise serves as an introduction to C/C++ and does not count towards admission to the final exam.
- Exercises are graded coarsely into only two categories: ***fail*** or ***pass***
    - if all the exercises (without considering the first introductory exercise) are completed successfully, then a bonus is carried over to the final exam[1]
- Turned in solutions will be scored based on functionality and readability
    - a solution which does not compile, run or produce a result will be a ***fail***
    - the amount of time spent on a solution cannot be taken into account for the score
- Exercises can be made and submitted in pairs
    - both partners must fully understand and be able to explain their solution
- Students may randomly be selected to explain their solution
    - details to be determined during exercises by the assistant

Exercise Submission Rules

- Submitted code must compile without errors.
- Code must compile and link either on Mac OS X or on a Unix/Linux/Cygwin environment using a Makefile
- The whole project source code (including Makefile) must be submitted via OLAT by the given deadline. Include exactly the files as indicated in the exercise.
- We will use MOSS to detect code copying or other cheating attempts, so write your own code!
- The whole project (including Makefile) must be zipped, and the .zip archive has to be named: ss_ex_EXERCISENUMBER_MATRIKELNUMBER.zip (ss_ex_3_01234567.zip - one student, Systems Software exercise number 3; ss_ex_1_01234567_02345678.zip - two students, Systems Software exercise number 1).
- Submit your code through OLAT course page.
- Teaching Assistant: Claudio Mura (claudio@ifi.uzh.ch)

Help with C++

Moving from Java to C++:	http://horstmann.com/ccj2/ccjapp3.html
C++ Tutorial:	http://www.cplusplus.com/doc/tutorial/
C++ Library Reference:	http://www.cplusplus.com/reference/
C++ STL:	http://www.sgi.com/tech/stl/table_of_contents.html
C++ FAQ Lite:	http://www.parashift.com/c++-faq-lite/

[1] the bonus will be in the order of 10% of the total number of points achievable in the final exam

NOTE: this exercise requires that you are familiar with some of the features of C++ (in particular I/O streams, templates and STL). Make sure you have acquired sufficient knowledge of these topics before you try to solve this exercise.

1. Introduction to C++

In this exercise you should read from an input text file the content of the personnel database of a small company. You should then sort the entries in ascending order according to the annual salary and write them to an output file in the new order.

The name of the input file should be passed as command line argument to your C++ application. The file will consist of *n* lines, one for each employee. A line is composed of 5 space-separated fields, which are respectively surname, name, yearly salary, age, clearance level. An example of a valid input file is given below:

```
Hicks Mike 45500 47 1
Smith Kyle 32000 57 3
Jones Catherine 27000 32 4
```

You should open the file using the `std::ifstream` class provided by the C++ Standard Template Library (STL) and create a struct object for each employee record you read. The structs objects should be stored in a `std::vector`, which should then be appropriately sorted by salary using the `std::sort` function in the STL. To do this, you should create a *functor* and pass it to the sorting function.

The contents of the sorted vector should be written out to a text file named `sorted_db.txt` using the `std::ofstream` class. The format of this output file should be the same as that of the input file.

Notes:

- you are provided with a sample Makefile that builds a simple application; you can adapt this Makefile and use it for your project

Deliverables:

Electronically submit your project files (i.e. source files and Makefile) and a README file that briefly explains your program, all in a single ZIP file.

Deadline: 27th September, 2015 at 23.59h.