4th International Conference on Higher Education Advances (HEAd'18)
Universitat Politècnica de València, València, 2018
DOI: http://dx.doi.org/10.4995/HEAd18.2018.8176

Accessible C-programming course from scratch using a MOOC platform without limitations

Castelló, Adrián[a]; Iserte, Sergio[a]; Belloch, Jose A.[b]
[a]Dpto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I, Castelló (Spain).
[b]Dpto. de Tecnología Electrónica, Universidad Carlos III de Madrid (Spain).

Abstract

The C language has been used for ages in the application development in multidisciplinary environments. However, in the academia, this language is being replaced by other higher-level languages due to they are easier to understand, learn and apply. Moreover, the necessity of professionals with a good knowledge in those high-level languages is constantly increasing because of the boosting of mobile devices. This scenario generates a lack of low-level language programmers, required in other less trendy fields, but equal or more important, such as science, engineering or research.

To revive the interest in low-level languages and provide those minority fields with well-prepared staff, we present in this work a C-programming massive online open course (MOOC) that is addressed to any kind of people with or without IT background. A feature that differentiates this course from others programming online-based courses is that we mainly focus on the C language syntax providing, via a self-tuned virtual machine, an encapsulated environment that hides any interaction with the command-line of the underlying operating system. A secondary target of this work is to foster the computer science degree students to enrol the computer architecture specialization at the Universitat Jaume I (Spain). For this purpose, the High-Performance Computing and Architectures research group of that University has decided to use this C course as a tool for fulfill the gap of the current syllabus.

The results show that half of the participants that completed the first session of the course have satisfactorily finished the course, and the number of computer science degree students that chose the computer architecture specialization the following academic course was increment by 3x.

Keywords: *MOOC; C language; Programming course*

1. Introduction

Engineering students often have difficulties with learning a programming language. Programming difficulties were identified by Butler and Morgan (2007), and Bosse and Gerosa (2017). To the usual difficulties of learning programming for the first time (a slightly different way of thinking is required), we need to add that C is one of the most difficult programming languages to learn, requiring the knowledge of several new and unique to C concepts with a slow learning curve, meaning that before writing the first program, students have to learn and understand different concepts, such as types of variables, functions, memory allocation of variables, among others.

As a consequence, this language is being replaced in the academia by other higher-level languages due to they are easier to understand, learn and apply. Moreover, the necessity of professionals with a good knowledge in those high-level languages is constantly increasing because of the boosting of mobile devices. This scenario generates a lack of low-level language programmers, required in other less trendy fields, but equal or more important, such as science, engineering or research.

To revive the interest in low-level languages and provide those minority fields with well-prepared staff, we present in this work a C-programming MOOC that is addressed to any kind of people with or without IT background (see Klobas et al. (2015) and Guo et al. (2014)). Pawelczak and Baumann (2014) carried out a first approach by designing a Virtual-C IDE as programming environment for beginners. A feature that differentiates this course from others programming online-based courses is that we mainly focus on the C language syntax providing, via a self-tuned virtual machine, an encapsulated environment that hides any interaction with the command-line of the underlying operating system.

The course is aimed to the practice since the very beginning. It is composed of 5 sessions, each one containing a set of videos with the explanations, a self-contained hands-on activity, and an auto-evaluated test. The design of the course followed the guidelines of the works presented by Shyr (2010), Bonwell (1991) and McKeachie (1994), that indicate that students retain much more when they directly go to practice sessions in contrast to the traditional classroom lectures where they only see or listen concepts. Lahtinen et al. (2005) also find in their study, that most helpful for students' learning is writing programs on their own.

A secondary target of this work is to foster the Computer Science degree students to enrol the computer architecture specialization at the Universitat Jaume I (Spain). For this purpose, the High-Performance Computing and Architectures research group of that University has decided to use this C course as a tool for fulfill the gap of the current syllabus.

This article is structured as follows: Section 2 describes the concepts and the five sessions of the C-programming course. Section 3 offers the technical implementation of the MOOC. Section 4 offers statistical results attending that were collected from the participants. Finally, some conclusion remarks are devoted to Section 5.

2. Summary of the five sessions

We have divided our course in five sessions that cover all tools that a participant without any C-knowledge could require in order to quickly develop a simple program. Figure 1 shows the MOOC welcome page from the course where all the materials, including the links to the videos, are still available.

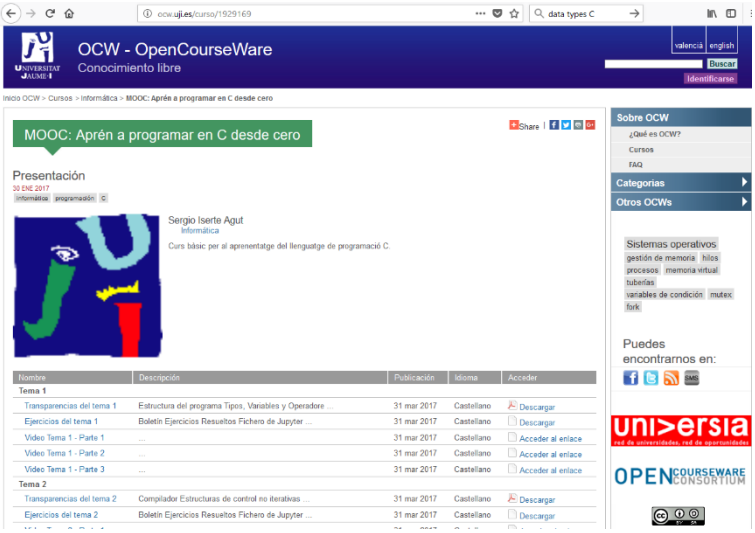

Figure 1: URL from the MOOC course: http://ocw.uji.es/curso/1929169

The five sessions of the course cover the following aspects of the C-programming:

Session 1: Data Types

We present in this session the four basic arithmetic type specifiers *char*, *int*, *float* and *double*, and the modifiers *signed*, *unsigned*, *short* and *long*.

Session 2: Non-Iterative control structures.

Session 2 is devoted to the sentences that allow to decide which part of the program must be executed in each moment. We focus mainly on the sentences *if-else* and *switch*.

Session 3: Iterative control structures.

Iterative control structures collects the group of sentences that allow to repeat different parts of the program. In case the number of iterations is known, we use the stament *for*. In other cases, we use the staments *while()*, and *do … while ()*.

Session 4: Vector, Chains and Functions

Session 4 focuses on the definition of static memory arrays, such as: vectors, matrices and chains of characters. Moreover, the participants of the course will also learn how to design functions and how they interact with the rest of the code.

Session 5: Memory Issues

The last session is the most difficult one and describes the memory access when complicated data structures are used. Participants learn to deal with the complexity of pointers.

3. Workplace deployment

Aware that this course was addressed to people without any previous knowledge in programming, we decided to avoid any interaction between the participant and the system console or the command line. For this purpose, we deployed a ready-to-use virtual machine with all the needed software. Figure 2 illustrates the setup that participants were expected to have in their own workstations.

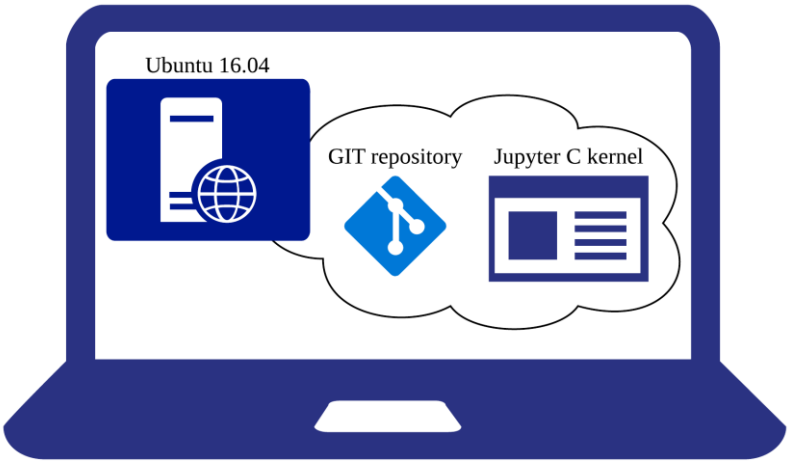

Figure 2: Setup that students were expected to have in their own workstations.

We based our platform on a GNU/Linux Ubuntu 16.04, installed in a VM. In order to handle the VM, we chose Oracle VM VirtualBox mainly because its acceptance in desktop environments and its easy installation, in any platform, for non-experienced users.

Apart from the native software included in Ubuntu, the VM came with a GIT repository of the course with the files loaded by Jupyter in each session. This local repository was weekly synchronized (when a new session was released) with one in BitBucket[1]. A script automatically executed when the user logged in the system was in charge of this update.

The main tool for developing and executing the codes, was Jupyter. Particularly, a version of Jupiter that includes a C kernel[2] which allows the users to compile and execute with only pressing [Ctrl + Intro] in their keyboard.

4. Results

First result consists on evaluating the number of participants that completed each one of the sessions, see Figure 3. It is important to point out that half of the students that signed out for the first session finished successfully the course.

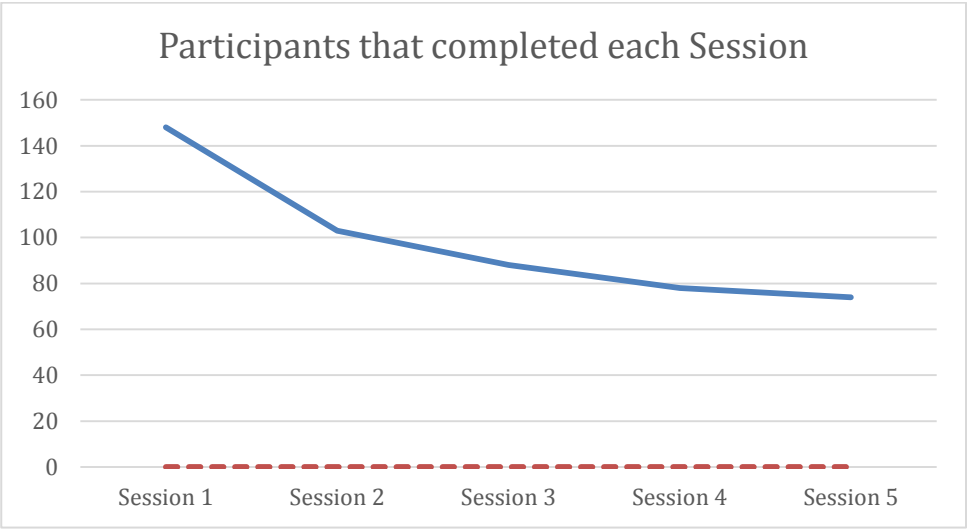

Figure 3: Number of participants that completed each session.

Afterwards, we ask the participants that assess the complexity of the sessions, as well as the number of hours that they required in order to complete each one of the sessions. Figures 4 and 5 collect information of this kind. As it is appreciated in Figure 4, the time employed

[1] https://bitbucket.org/
[2] https://github.com/brendan-rius/jupyter-c-kernel

by the participants for performing the first two sessions is around 1-2 hours, which is reasonable since they deal with the most basic concepts. As the course advances, the percentage of participants that require 3-4 hours increases, being the session 5 the one that agglutinates the most percentage of participants that require more than 5 hours. Note that in average, most of the participants require 3-4 hours to complete each session.

Regarding the complexity of the sessions that is shown in Figure 5, we appreciate that the first two sessions had a difficulty which could be *expected* by the participants, while sessions 3, 4 and 5 were considered as *difficult*, which is also a coherent result. It is important to highlight the low percentage of the participants that considered globally the sessions as *hard*.

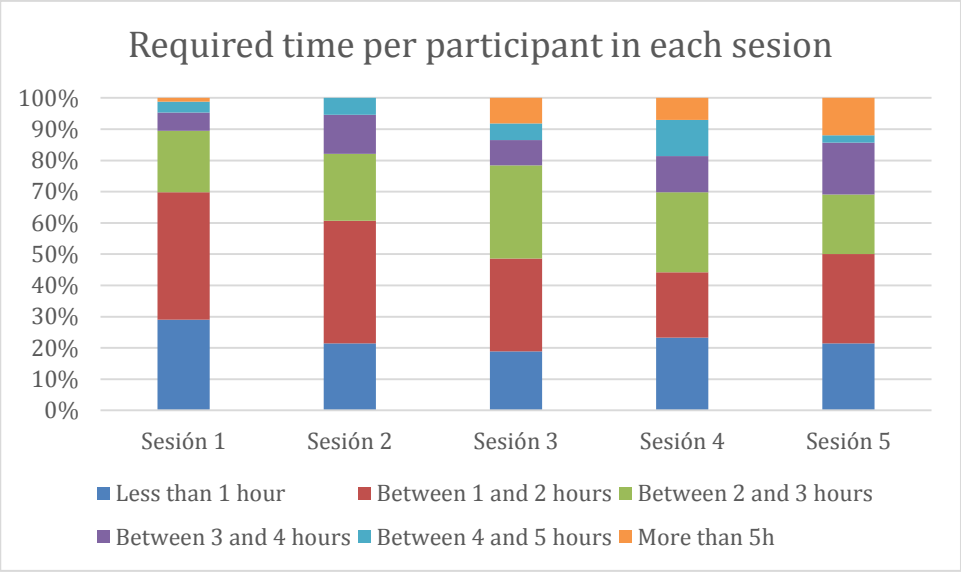

Figure 4: Percentage of participants together with the required time to complete each session.

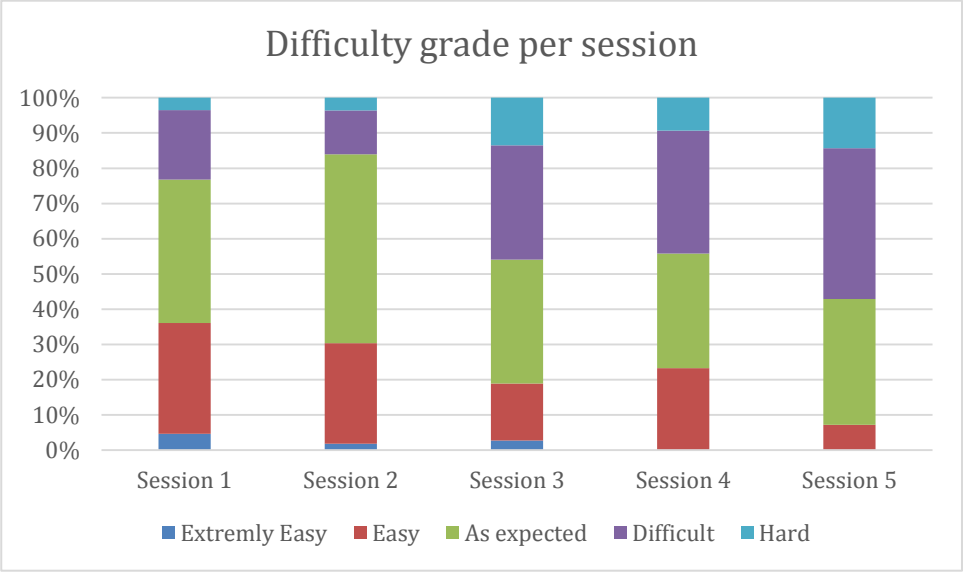

Figure 5: Percentage of participants together with the difficulty grade to complete each session.

Finally, this course aimed also to foster the computer science degree students to enrol the computer architecture specialization at Universitat Jaume I (Spain). As a result, we want to stand out that the number of computer science degree students that chose the computer architecture specialization the following academic after the first edition of the course was increased by three.

5. Conclusion

The first conclusion that we want to extract is that half of the participants that signed out for the first session finished successfully the course. Considering the surveys that were completed by the participants, we can consider that there is still features to improve but that there exists a global satisfaction for this course, which will continue being offered by the Universitat Jame I (Spain) for the following years.

Acknowledgements

This research has been partly funded by TIN2017-82972-R. Adrián Castelló was supported by the ValI+D 2015 FPI program of the Generalitat Valenciana.

References

Bosse, Y., & Gerosa, M. A. (2017). *Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage*.SIGSOFT Softw. Eng. Notes 41, 6 (January 2017), 1-6

Butler M. & Morgen, M. (2007). *Learning challenges faced by novice programming students studying high level and low feedback concepts*. In ICT: Providing choices for learners and learning. Proceedings ascilate Singapore 2007, 99-107.

Klobas, J. E., Mackintosh, B. & Murphy, J. (2015). *The Anatomy of MOOCS*. In: P. Kim (Ed.), Massive Open Online Courses. The MOOC Revolution. New York & London: Routledge.

Guo, P. J., Kim, J., & Rubin, R. (2014). *How video production affects student engagement: an empirical study of MOOC videos*. Proc. of the 1st ACM Conference on Learning, Atlanta, Georgia, USA. doi:10.1145/2556325.2566239.

Pawelczak, D. & Baumann, A. (2014): *Virtual-C – a programming environment for teaching C in undergraduate programming courses*. In Proc. of the IEEE Global Engineering Educ. Conf. (EDUCON), 3-5 April 2014, Istanbul, Turkey, 1142-1148

Bonwell, C. C. & Eison, J.A. (1991). *Active learning: Creating excitement in the classroom*, Washington C. C. George Washington University

Christensen, C. M., Horn, M. B. & Johnson, C. W. (2010). *Rethinking student motivation: Why understanding the 'job' is crucial for improving education*, Boston: Innosight Institute.

Mckeachie, W. J. (1994). *Teaching Tips: Strategies, Research, and Theory for College and University Teachers, 9th ed*. Lexington, MA: Cengage learning.

Lahtinen, E., Mutka, K. A., & Jarvinen, H. M. (2005). *A Study of the difficulties of novice programmers*, In Proc. of the 10th Annual SIGCSE Conf. on Innovation and Technol. In Comput. Sci. Educ. ITiCSE'05, 14-18