Improving Performance and Progression of Novice Programmers: Factors Considerations

Hala Shaari and Nuredin Ahmed

***Abstract*—Teaching computer programming is recognized to be difficult and a real challenge. The biggest problem faced by novice programmers is their lack of understanding of basic programming concepts. A visualized learning tool was developed and used by volunteered first-year students for two semesters. The purposes of this paper are: Firstly, to emphasize factors which directly affect the performance of our students negatively. Secondly, to examine whether the proposed tool would improve their performance and learning progression or not. This tool provides many features and enhancement which were presented to students as pre-lecture material. The results of adopting this tool were conducted using a pre-survey and post-survey questionnaire. As a result, students who used the learning tool showed better performance in their programming subject.**

***Index Terms*—Factors, novice, programming, visualization.**

I. Introduction

Novice students in universities find learning how to Program difficult due to simultaneous learning of syntax, semantic of programming as well as how to interpret error messages. Many studies examined the factors which may affect the performance of the programming. However, factors that are considered as powerful predictors of programming success had not been discovered yet. In addition, varies tools have been introduced to help novice programmers improve their performance. This study conducted an experiment of providing novice programmers with a pre-lecture Arabic tool to understand the concepts of programming languages prior to the lecture. This study was motivated by the poor performance and progression of our first-year C language course programming students. It was recognized that the language used in the programming was affecting our Students' performance badly, since their level of English does not allow them to understand the provided material perfectly.

Therefore, our goal was to provide our students with an Arabic Tool to introduce C language concepts which should be used as a pre-lecture tool to help them get the concepts prior to the lecture.

II. Background

Learning and teaching how to program is not an easy task. In recent years multi-national studies showed that students have problems in writing program codes [1], [2]. These problems are caused by the lack of correct understanding of abstract concepts. Which seems to be difficult for many students [3]-[7]. The factors that can influence success in programming have been investigated by many studies. Factors such as expertise in their spoken language, the number of used and analyzed programming languages, mathematical ability, previous academic degree, measures of general intelligence, self-confidence of students and gender have been extensively researched. According to these researches not all the listed factors affect the performances of programming. Factors that show strong positive correlation of success in programming are the ability to solve problems in other sciences like physics and mathematics [7]-[12] Prior knowledge of programming [12]-[14] and students with viable mental models [15]-[17]. However, other factors such as gender, experiences and familiarity of programming language did not show any significant effect on their performance [12]-[14]. In addition, one of the most challenging factors in many countries is English as not being the first language which affects the students' performance [14], [18], [19].

To help novices learn programming, several tools were developed. Narrative tools, visual programming tools, flow-model tools, specialized output realizations, and tiered language tools are the categories that the tools were divided into [20]. Visualization is advantageous for learning many programming concepts in computer science education, which is why they have been used for long period of time [21]. Visual programming allows programmers to construct a program without writing any code through a drag-and-drop interface (e.g., JPie, Alice, Scratch, Karel Universe). Alice increase performance rate and motivation programming according to researches [22]-[24]. Furthermore, Scratch is found to be an effective tool to introduce students to programming [25], [26]. We take in our considerations how the material could be delivered to the students. Few alternate delivery techniques for novice programmers are available in literature compared to alternate delivery modes in education. A research has suggested that students' comprehensions on introductory programming subject were improved after providing them with prerecorded mini-lectures [27].

III. Methodology

This section illustrates the data collection process and the learning environment of the study. The study's goal was to discover whether or not the proposed tool helped participants to improve their performance and progression through their

Manuscript received January 12, 2017; revised April 1, 2017.

Hala Shaari is with the Department of Software Engineering, Faculty of Information Technologies, University of Tripoli, Libya (e-mail: h.shaari@uot.edu.ly).

Nuredin Ahmed is with the Department of Computer Engineering, Faculty of Engineering, Azzaytuna University, Libya (e-mail: nuredin@mtit.com.ly).

doi: 10.18178/ijiet.2018.8.1.1003

first programming course. With the assumption that students had no valuable knowledge, the methodological basis for the research was designed.

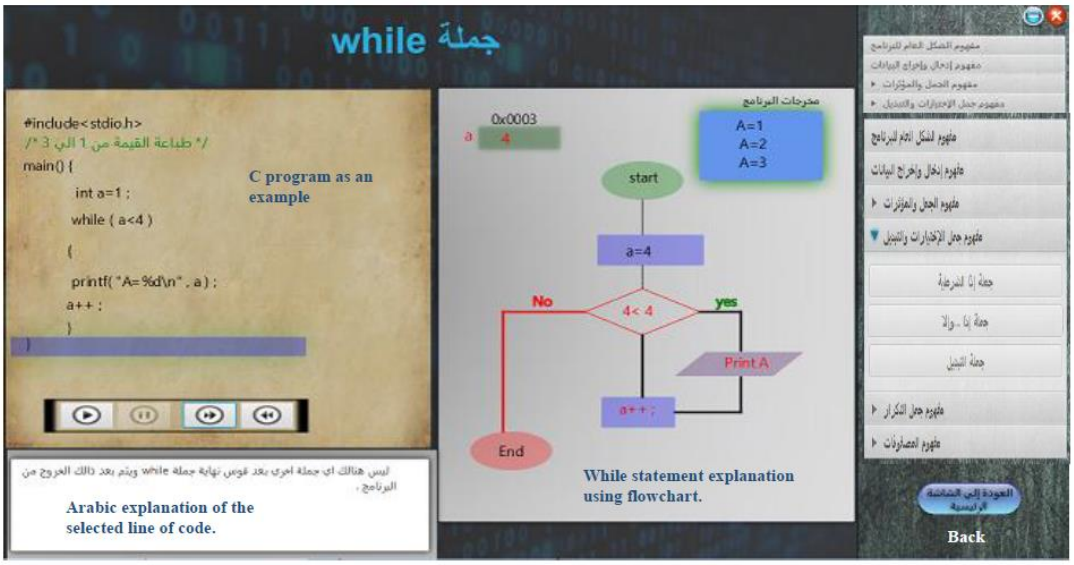

Fig. 1. The learning environment.

A. Data Collection

The dataset used in this study is gathered from the first year volunteered students of faculty of information technology at Tripoli university. The data collections were conducted in the fall semester in 2015 and the spring semester 2016. Data were collected using the methods below:

- A structured questionnaire used to survey students at the beginning of the semester. We referred to as pre – experiment survey.
- A proposed standalone learning environment was applied.
- An end of semester survey of students whom used the proposed tool, using a structured questionnaire. We referred to a post-experiment survey.

1) Survey of students

At the beginning of the two semesters a paper-based questionnaire was handed to registered students in introduction to programming course. Since this course was taken by most first year students registered in the IT degrees. One hundred students have responded to the survey. Most of the data were gathered during the first weeks of the semester. Students were informed of the motivations and objectives of both surveys.

The questionnaire includes closed response style questions. Many different domains have been considered when the questionnaire was designed, such as whether the students use assistance tools to help them understand the concepts, if they are struggling with the programming instructions since they are available in a different language and if they can imagine the execution process easily. In addition, after applying the proposed tool for participated students, different paper-based questionnaire was handed to inquire the efficiency ratio of the learning environment. The questionnaire represented a number of different domains such as if this tool helped them to understand the provided concepts, whether the techniques used in this tool improve their understanding and if the provided examples and exercises assisted them to improve their learning progression.

B. The Learning Environment

A proposed learning tool implemented using JavaFX was introduced and applied for volunteered participants. It is providing many features and enhancements that can help to develop rich applications. This learning tool currently supports C language primary concepts, but in the future we may support other languages. The dynamic execution processes of a program are visualized and simulated by this visualization tool (Fig. 1).

Many concepts were provided by this tool such as assignment, if statements, loops, switch statement and arrays. Also, students are able to run the program gradually. This means that students have the chance to control the speed and would be able to watch and understand what happens when one statement is executed. This function is very important to the visualization tool to help improve the student's understanding. In addition, graphical representation and animation have been used to visualize the dynamic execution with every statement's execution. Animation is used to leverage the benefits of the proposed tool. Animation has great potential significance to help students to improve their understanding of programming concepts. Furthermore, students were able to repeat the program execution as well as pause, resume and stop it. That gave them the opportunity to determine and correct their misunderstanding of the concepts. Meanwhile, Arabic textural explanations also are provided to explain the execution process of each statement. To obtain the best results from our experiments, the proposed tool is provided with another two sections to help participants to receive better understanding. The first section explained the primary concepts of the C language in particular order presented as textual context. However, the concept explanation is sorted according to a text book which was used by the course lecturers. Each concept is supported with the visualized dynamic execution process as explained above as

well as two to three exercises for each concept (second section) ,So that participants could test their understanding for each concept. For instance, the NOT operator concept is explained as a textual context (Fig. 2). After reading this explanation, Participants have the choice to move either to the visualized dynamic execution process of NOT operator concept (Fig. 3) or to their associated exercises.

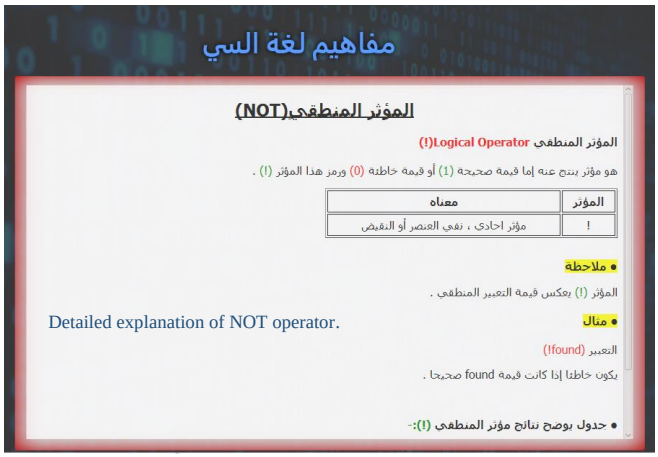

Fig. 2. NOT operator concept textual explanation.

IV. Results

Descriptive information was provided by analyzing students' survey responses. The SPSS software was used to preform descriptive and statistical analysis of the quantitative data.

A. Pre-experiment Survey Results

The survey results revealed that 69% of participated students found programming module difficult to understand. These difficulties are resulted from their failure to imagine the execution process of programming, 65% claims. Also, 77% of participated are seeking other ways to improve their performance. Therefore, 98% of them approved utilizing tools to improve their programming concept understanding. However, 67% of the students preferred the tools to be Arabic to improve their understanding. The overall finding suggested the need of assistance tools to better learning progression as showed in (Fig. 4).

B. The Experiment

After collecting data in the pre–experiment questionnaire stage, the participants went on to use the visualization tools. The participants were asked to execute program fragments which were explained by the proposed tool visually.

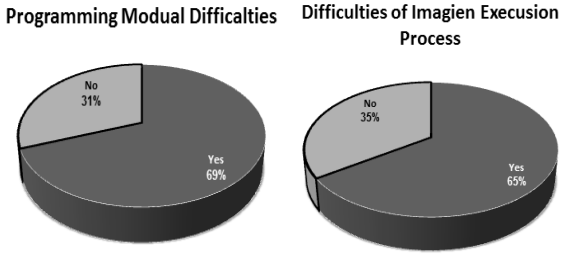

Fig. 4. Some pre- experiment survey results.

Simple concepts were explained as well as some relatively hard concepts (switch case, arrays). These concepts were explained in many different techniques for instance flowchart used to visualize the while loop execution process (figure5). These explanations were provided to participants as pre-lecture material. However, these concepts were supported by many different examples and exercises for better understanding.

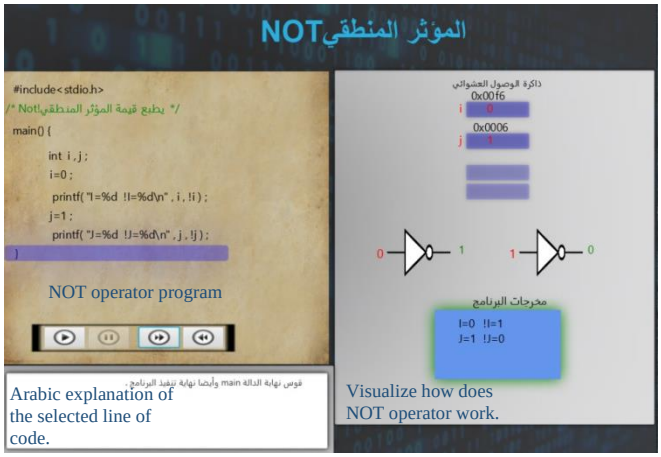

Fig.3 visualized dynamic execution process of NOT operator concept.

C. Post-experiment Survey Results

A post-experiment questionnaire was also used to collect quantitative data along with the experiment. Three significant attributes have been discovered of the teaching environment from the survey's feedback. Firstly, participants pre-existing understanding of the concept could be changed by the provided animated execution. Secondly, it was considered for the animations to be very helpful in increasing the concepts' understanding. Finally, the gradual execution was seen as other useful feature. Moreover, the Arabic explanation of the concepts provided by the proposed tool leverage the benefits of using this tool. According to survey results, there was an improvement of participant's understanding, with 70% of participants approving the benefits of using this tool through their module results.

V. Discussion and Conclusion

The motivation of this experiment was due to the high failure in our first-year programming module. Many factors have been considered to influence programming. Some factors that might predict success in programming were usually observed factors such as expertise in their spoken language, gender, mathematical ability. However, it was recognized that the language used in the programming affects our Students' performance sorely. The abilities of the proposed tool which includes textual concepts explanation, visualized dynamic execution and exercises increases the possibilities of participants' better understanding. Textual explanation is used to offer an extra explanation for each concept. However, participants were able to examine their understanding of a single concept by using the exercises which associated every concept.

The positive role that visualization plays is, revealed by this study, improving students' performance. While the visualization tool was useful in improve the participants' understanding of the covered concept successfully, it is

proved that the accompanied Arabic explanation helped to improve concepts understanding which leaded them to enhance their performance and learning progression. Students who used the tool showed improvement in their performance according to their course assessments. The results of this initial study suggest that, using our tool as a pre-lecture tool helped our participants to better understanding of programming concepts prior to the lecture. However, investigating the effectiveness of the proposed tool in actual pedagogical context would make results even more accurate.

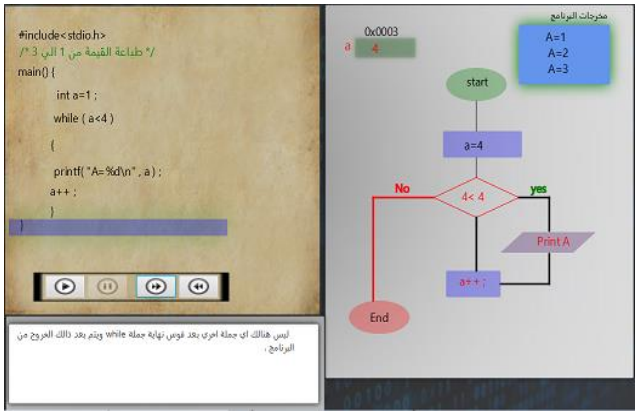

Fig. 5. Flowchart used to explain while loop execution process.

REFERENCES

[1] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students," *ACM SIGCSE Bulletin*, 2001, vol. 33, no. 4, pp. 125-140.

[2] J. Vegso, "Interest in CS as a major drops among incoming freshmen," *Computing Research News*, vol. 17, no. 3, pp. 126-140, October 14, 2009.

[3] C. Mow," Issues and difficulties in teaching novice computer programming," *Innovative Techniques in Instruction Technology, e-Learning, e-Assessment, and Education*, Netherlands, pp. 199-204.

[4] E. Lahtinen, K. Ala-Mutka, and H. Järvinen. "A study of the difficulties of novice programmers," in *Proc. the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 2005, Portugal, pp. 14-18.

[5] A. Robins, J. Roundtree, and N. Roundtree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137-172, September 14, 2008.

[6] D. Sleeman, "The challenges of teaching computer programming," *Communications of the ACM*, vol. 29, issue 9, Sept. 1986, pp. 840-841.

[7] E. M. Soloway, "Learning to program = Learning to construct mechanisms and explanations," *Communications of the ACM*, vol. 29, issue 9, Sept. 1986, pp. 850-858.

[8] P. Byrne and G. Lyons, "The effect of student attributes on success in programming," *ITiCSE 2001*, 2001.

[9] G. White and M. Sivitanides, "An empirical investigation of the relationship between success in mathematics and visual programming courses," *Journal of Information of Systems Education*, vol. 14, no. 1, pp. 409-416, 2003.

[10] J. Bennedsen and M. Caspersen, "An investigation of potential success factors for an introductory model-driven programming course," in *Proc. of the 1st. Intl. Computing Education Research Workshop*, ICER 2005, 2005.

[11] N. Pillay and V. R. Jugoo "An investigation into student Characteristics affecting novice programming performance," *ACM SIGCSE Bulletin*, vol. 37, no. 4, pp. 107-110, December 2005.

[12] N. Bubica and I. Boliat, "Predictors of novices programmers' performance," in *Proc. the ICERI2014 Conference*, November 2014, Seville, Spain.

[13] K. Paivi and S. Beth, "My program is OK - am I? Computing freshman's experience of doing programming assignments," *Computer Science Education*, vol. 22, no. 1, pp. 1-28, March, 2014.

[14] J. Sheard, A. Carbone, S. Markham, A. J. Hurst, D. Casey, and C. Avram, "Performance and progression of first year ICT students," in *Proc. 10th Australian Computing Education Conference-ACE 2008*, Wollongong, Australia.

[15] L. Ma, "Investigating and improving novice programmers' mental models of programming concepts," University of Strathclyde, Department of Computer & Information Science, 2007.

[16] L. Ma *et al*., "Improving the mental models held by novice programmers using cognitive conflict and jeliot visualisations," presented at ITiCSE'09, Paris, France, 2009.

[17] R. Bornat, S. Dehnadi, and Simon, "Mental models, consistency and programming aptitude," in *Proc. the Tenth Australasian Computing Education Conference (ACE2008)*, Wollongong, Australia, January 2008.

[18] D. W. Edsger, "How do we tell truths that might hurt?" *Computing: A Personal Perspective*, Springer-Verlag, pp. 129–131, 1982.

[19] M. Elisapeta and T. Edna, "Exploratory study on factors that impact / influence success and failure of students in the foundation computer studies course at the National University of Samoa," *Journal of Emerging Trends in Computing and Information Sciences CIS Journal*, vol. 3, no. 5, May 2012.

[20] K. Powers, P. Gross, S. Cooper, M. McNally, K. J. Goldman, and V. Proulx, "Tools for teaching introductory programming: What works?" in *Proc. the 37th SIGCSE Technical Symposium on Computer Science Education*, USA, pp. 560-561, 2006.

[21] C. Yehezkel, M. Ben-Ari, and T. Dreyfus, "Computer architecture and mental models," *ACM SIGCSE Bulletin*, 2005, vol. 37, no. 1, pp. 101–105.

[22] C. Bishop-Clark, J. Courte, D. Evans, and E. Howard "A quantitative and qualitative investigation of using Alice programming to improve confidence, enjoyment, and achievement among non-majors," *Journal of Educational Computing Research*, 2007, vol. 37, no. 2, pp. 193-207.

[23] S. Cooper, W. Dann, and R. Pausch "Teaching objects-first in introductory computer science," in *Proc. the 34th SIGCSE Technical Symposium on Computer Science Education*, USA, pp. 191-195, 2003.

[24] J. Courte, E. Howard, and C. Bishop-Clark, "Using alice in a computer science survey course," *Information Systems Education Journal*, vol. 4, no. 87, pp. 1-7, 2006.

[25] D. J. Malan and H. H. Leitner, "Scratch for budding computer scientists," *ACM SIGCSE Bulletin*, 2007, vol. 39, no. 1), pp. 223-227.

[26] P. A. Sivilotti and S. A. Laugel, "Scratching the surface of advanced topics in software engineering: a workshop module for middle school students," in *Proc. the 39th SIGCSE Technical Symposium on Computer Science Education*, 2008, USA, pp. 291-295.

[27] G. Smith and C. Fidge, "On the efficacy of prerecorded lectures for teaching introductory programming," *Tenth Australasian Computing Education Conference (ACE2008)*, vol. 78, Simon and Margaret Hamilton, Ed, Australian.

H. Shaari was born in Alen, Germany in July 1976. She studied the BSc. in computer science at University of Tripoli in 1997, Libya; the MSc. in computers and interment technologies from University of Strathclyde, UK in 2009.

She is currently lecturer at Faculty of IT at Uni. Of Tripoli. Her current research interest includes learning technologies and e-learning. She has extensive experience in design, programming of mobile system related apps.

N. Ahmed was born in Trhona in May 1971, Libya. He studied the BSc. in computer engineering at University of Tripoli in 1995, Libya; the MSc. in technical informatics from TU Clausthal, Germany in 2002; the PhD in electronics and electrical engineering from Glasgow University UK in 2011. He is currently an assistant professor of Computer Engineering Dept. At Azzaytuna University, Libya. His research interest includes embedded systems and modeling of communication networks using SystemC/C/C++ and Opnet.