e-ISBN : 978-967-25608-3-8

SIG : e-Learning@CS
https://appspenang.uitm.edu.my/sigcs/
Publication Date : 15 – Apr - 2022

PROGRAMMING RELATED FACTORS INFLUENCING IN LEARNING A PROGRAMMING: A STUDY ON ENGINEERING STUDENTS IN UITM PULAU PINANG

*Rozita Kadar[1], Jamal Othman[2], Naemah Abdul Wahab[3],
Maisurah Shamsuddin[4] and Siti Balqis Mahlan[5]
** rozita231@uitm.edu.my[1], jamalothman@uitm.edu.my[2], naema586@uitm.edu.my[3], maisurah025@uitm.edu.my[4], sitibalqis026@uitm.edu.my[5]*

[1,2,3,4,5]Faculty of Computer and Mathematical Sciences
Universiti Teknologi MARA, Cawangan Pulau Pinang, Malaysia

ABSTRACT

In response to a current need, higher education institutions have begun to provide programming courses not only to computer science students, but also to non-computer science students. This is because, in recent years, understanding of computer technology and programming in a variety of areas has become necessary to meet the industry's demand for information and communication competency. The issue is that teaching and learning computer programming languages, particularly for non-computer science students, can be difficult. Educators face challenges in getting students to comprehend programming principles and develop outstanding programming abilities in order to tackle real-world problems. Students' lack of logical, creative, and critical thinking leads to inadequacies in problem-based learning implementation. Several factors have been identified as contributing to programming language problems, including educators, students' abilities, and programming nature, and this study investigates a programming-related factor that contributes to computer programming learning challenges, as well as the students' background knowledge. A study was conducted on engineering students in Universiti Technologi MARA in a total of 241 students involves in this study. According to the results of a mean and standard deviation study, it is found that programming-related factor were particularly effective in learning a programming. With these findings, it can be a guideline for educators in dealing with problems in learning a programming.

Keywords: Programming Problem, Programming Nature, Program Structure, Engineering Student.

Introduction

Nowadays, having a good understanding of computer technology, as well as programming skills, is required to satisfy industrial demands (Siti Rosminah & Ahmad Zamzuri, 2012). For many students, particularly those with a non-computer science background, the current demand causes challenges and presents a considerable obstacle. According to (Moström, 2011), novice students must understand the problem, develop a solution using normal problem-solving methodologies, and then write down the solution in a programming language in such a way that a computer can understand the instructions.

According to the existing study, traditional teaching practices, as well as students' study methods and attitudes, must be improved. Instructional methods such as hands-on programming practise and cutting-edge teaching and learning tactics must be utilised to increase students' interest in computer education. Educators are faced with new challenges, demanding the creation of new

instructional tools. Students' background knowledge and attitudes, teaching and learning methods, and social context are all aspects that contribute to learning obstacles in computer programming, according to (Gomes et al., 2012). Educators must also address gaps in students' knowledge backgrounds, making large-scale student management more difficult (Ahmad & Ghazali, 2020).

Therefore, the purpose of this study is to identify the challenges that students have when learning a programming language by investigating the programming-related factors that influence students' programming language learning and proposing ways to overcome these concerns. This research is intended to help computer science educators improve their teaching methods for basic programming courses, as well as improve students' interest in and performance in programming courses.

Programming Nature in Learning a Program

Learning any programming languages is far more complicated, as it necessitates other abilities such as algorithm design, programming writing, and syntax understanding (Baist & Pamungkas, 2017). It is not easy to write a program code, according to (Moström, 2011), novice students must comprehend the problem, design a solution using standard problem-solving approaches, then write down the solution in a programming language in a way that a computer can follow the instructions.

The problem arise among students starting at the beginning is related to the understanding of programming environment, which causes students to see programming as something difficult (Ahmad & Ghazali, 2020). Also, the inability of students to reason logically and their lack of problem-solving skills are two major factors that contribute to programming inefficiency. Although different programming methods and approaches have been developed to assist students in learning programming, not all of them focus on the programming stages of problem resolution (Yusoff et al., 2020). Similarly, despite the existence of a variety of learning aids and teaching strategies, such as teaching by doing, using relevant examples, demonstrations, direct examples, and trail-guided teaching approaches, teaching issues and programming learning remain unresolved (Cheah, 2020).

One of the issues discussed regarding the effectiveness of students mastering a programming language is the nature of programming, which plays an important role in determining the effectiveness of students' ability to master a programming language. Pears et al. (2007) reported that most institutions use an object-oriented language, but many use Java, C and C ++, languages to teach procedural programming, whereas less than 10% of institutions teach functional programming.

Despite the popularity of such languages, there has been much debate about the suitability of these languages for education, especially when introducing programming to novices. These languages are not designed specifically for educational purposes, in contrast to others designed with this specific purpose (such as Python, Logo, Eiffel, and Pascal).

There are interrelated types of programming's' nature of difficulties while learning to program as stated in the previous studies. (Siti Rosminah & Ahmad Zamzuri, 2012) identified three issues that should be addressed by educators, which are: the lack of understanding of the basic concepts of programming structure; problem in designing a program to complete a specific task and; inability to identify the syntax of programming languages. This is in contrast to (Bosse & Gerosa, 2017), which is more focused on the ability of students in using computers and performing system development tasks. The present discussion focuses more on the opinions outlined by (Xinogalos, 2016), which has outlined five problems faced by students related to programming nature, namely: developing an algorithm, transferring an algorithm to a programming language, programming structures, modularisation, and; testing and debugging. Guided by (Xinogalos, 2016), these five issues were discussed based on the study on previous work as well as observations of more than 10 years in the world of programming education. The findings of the present observation are stated in the summary as shown in Table 1.

The following section will discuss the survey conducted on non-computer science students that focuses on the factors that effect on learning a program. This focused factor is related to the nature of the program which will show the environment of a programming language in giving effect in the programming learning process.

Table 1 Programming-Related Factors

Problem	Author(s)	Descriptions
Basic knowledge of Programming	(Chan Mow, 2008; Costa et al., 2012; Lahtinen et al., 2005; M, 2014; Xinogalos, 2016)	• Unable to transform the problem into a programming instruction. Most students may understand the syntax and semantics of individual statements, but they have no idea how to put them together into legitimate programmes. • Difficulties with language libraries, such as looking through them, finding the right function, and correctly using it in a programme. • Inability to combine syntax, logic, and concepts. Insufficient ability to translate problems into a charitable action plan. • A lack of understanding of effective instructional methods. • Difficulties in representing a program using notation. The symbols of a programming language, as well as the grammatical rules for assembling them into a programme, are referred to as notation.
Understanding the Structure of Programming	(Bosse & Gerosa, 2017; Chan Mow, 2008; Lahtinen et al., 2005; Qian & Lehman, 2017; Swidan et al., 2018; Wittie et al., 2017; Xinogalos, 2016)	• Failure to recognise that each command is carried out in the state generated by the preceding ones. • The complexity of comprehending the order of statements, the value of a variable, and the interactivity of an input action. • The most challenging programming concepts are pointers, arrays, and data structures.
Module Structure of Programming	(Bosse & Gerosa, 2017; Xinogalos, 2016)	• Students' difficulties in working with functions. • A lack of understanding of the scope of variables and why passing and returning arguments is required.
Testing and debugging	(Bosse & Gerosa, 2017; Chan Mow, 2008; Pears et al., 2007; Qian & Lehman, 2017; Siti Rosminah & Ahmad Zamzuri, 2012)	• Incapable of mastering the compiler, as well as error and warning messages • One of the difficulties is dealing with syntax mistakes, with the most typical issue being a lack of ability to discover faults. • Inability to visualise the status of the program during code execution. • Missing semicolons, mismatched parentheses, brackets, or quotation marks, and employing the illegal start of expressions.

Methodology

This study involved a total of 241 students who took programming courses at UiTM Cawangan Pulau Pinang. It consists of diploma and degree students from the Faculty of Mechanical Engineering (FKM) and the Faculty of Civil Engineering (FKA) as shows in Figure 1. Students are required to answer all questionnaires related to this course after they have completed the 14-week lecture.

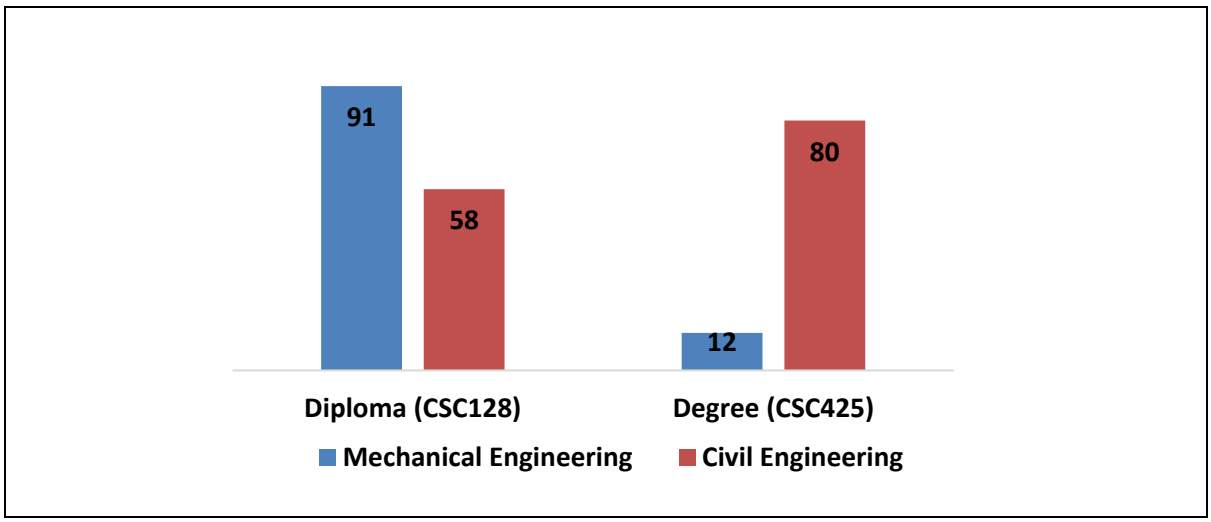

Figure 1. Number of Students in the Study

The questionnaire for the study consists of 2 parts as shows in Table 2 and Table 3. The first part is about the course information taken by students and the second part focuses on the factors related to the students on the programming course. All questions contain 15 items that focused more on Basic Element of Programming (4 items), Control Structure of Programming (5 items) and the Module Structure of Programming (4 items). These questionnaires use the five-point Likert scale. the programming ability-related factors used range from 5-Strongly Agree, 4-Agree, 3-Natural, 2-Disagree, and 1-Strongly disagree that values greater than 3 are positive and values less than 3 are negative statements.

Table 2 Construct Questions in Course Information

Construct	Options
Program Code	1. Mechanical Engineering 2. Civil Engineering
Study Level	1. Diploma 2. Degree
Semester	1/2/3/4/5/6/7/8
Programming Code Taken	1. CSC128 2. CSC425
Status Taken	1. First Timer 2. Not First Timer

Table 3 Construct Questions in Programming-Related Factor

Construct	Statements
A. Basic Element of Programming	1. Has the ability to define identifier. 2. Has the ability to write declaration statement. 3. Has the ability to assign value into variable. 4. Has the ability to evaluate/trace the mathematical expression based on certain input.
B. Control Structure of Programming	1. Has the ability to write a selection control structure. 2. Has the ability to write a repetition control structure. 3. Has the ability to trace a selection control structure program segment and show the output based on the given input. 4. Has the ability to trace a repetition control structure program segment and show the output based on the given input. 5. Has the ability to write a complete program that includes the combination of selection & repetition control structure.
C. Module Structure of Programming	1. Has the ability to apply the predefined function in a program. 2. Has the ability to apply the user-defined function in a program. 3. Has the ability to apply the local and global variables. 4. Has the ability to determine appropriate function either function with passing/return value or without passing/return values.

Reliability Test or Cronbach's Alpha was performed first before analyzed the questionnaire. Reliability describes how reliable and consistent a research instrument's measurement of a variable is. The better the instrument's reliability, the less errors it generates (Kumar, 2018). Cronbach's Alpha values are based on (Choi et al., 2001).

Cronbach's Alpha is used in this analysis to measure the internal consistency of the items tested. According to Table 4, the Cronbach's Alpha values for all 15 questionnaires tested was 0.935. This value is greater than 0.8, which is considered reliable.

Table 4. Reliability Test

Cronbach's Alpha	N of items
.935	15

Analysis and Result

In this paper, the mean and standard deviation was applied. The analysis by using mean and standard deviation values can be used to identify in general about programming ability-related factor that influence students in learning a programming language. The findings in Table 5 for programming abilities indicate that the mean for all items is near to or higher than 4. The ability of students to construct a declaration statement has the greatest mean of 4.11, while the ability to determine an appropriate function, either function with return or without return values, has the lowest mean of 3.55. The standard deviation is also not very high. This indicates that all students agreed to have the ability to understand the basic elements, control structure, and structure of programming modules.

Table 5. The Mean and Standard Deviation for Programming Ability- Related Factors

Item no.	Statement	Mean	SD
C1	Has the ability to define identifier.	4.04	0.679
C2	Has the ability to write declaration statement.	4.11	0.665
C3	Has the ability to assign value into variable.	4.06	0.677
C4	Has the ability to evaluate/trace the mathematical expression based on certain input.	3.94	0.725
C5	Has the ability to write a selection control structure.	3.82	0.689
C6	Has the ability to write a repetition control structure.	3.69	0.711
C7	Has the ability to trace a selection control structure program segment and show the output based on the given input.	3.71	0.746
C8	Has the ability to trace a repetition control structure program segment and show the output based on the given input.	3.66	0.731
C9	Has the ability to write a complete program that includes the combination of selection & repetition control structure.	3.64	0.784
C10	Has the ability to apply the predefined function in a program.	3.68	0.703
C11	Has the ability to apply the user-defined function in a program.	3.66	0.689
C12	Has the ability to apply the local and global variables.	3.62	0.721
C13	Has the ability to determine appropriate function either function with passing/return value or without passing/return values.	3.55	0.763

Overall, it was found that programming related factors were very helpful in programming learning. This is because the mean is 3 and above means that students agreed with all statement.

Conclusion

Reading program source code is not same as reviewing ordinary documents and using textual representation as the major source of information causes several challenges in program comprehension. Although many methods and tools have been proposed to represent source code, experience have shown that textual presentation is the most suitable to represent a program. However, the problems still exist if the source code is used in a form of text-based due to the source code. Therefore, the challenges that have emerged in learning a program due to the nature of the programming language are discussed in this work. The goal of this study is to identify the challenges that students have when learning a programming language by investigating the elements that influence students' programming language learning and offering techniques for dealing with these concerns. This research is intended to help computer science educators improve their teaching methods for basic programming courses, as well as raise students' interest in and performance in programming disciplines.

References

Ahmad, S., & Ghazali, J. (2020). Programming Teaching and Learning: Issues and Challenges. *Fstm.Kuis.Edu.My*,*16*(1),724–398. http://fstm.kuis.edu.my/icits/2020/eproceeding/assets/files/ITS 033.pdf

Baist, A., & Pamungkas, A. S. (2017). Analysis of Student Difficulties in Computer Programming. *VOLT :JurnalIlmiah Pendidikan Teknik Elektro*, *2*(2), 81. https://doi.org/10.30870/volt.v2i2.2211

Bosse, Y., & Gerosa, M. A. (2017). Why is programming so difficult to learn? *ACM SIGSOFT Software Engineering Notes*, *41*(6), 1–6. https://doi.org/10.1145/3011286.3011301

Chan Mow, I. T. (2008). Issues and difficulties in teaching novice computer programming. *Innovative Techniques in Instruction Technology, E-Learning, E-Assessment, and Education*, 199–204. https://doi.org/10.1007/978-1-4020-8739-4-36

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, *12*(2), 1–14. https://doi.org/10.30935/cedtech/8247

Choi, N., Fuqua, D. R., & Griffin, B. W. (2001). Exploratory analysis of the structure of scores from the multidimensional scales of perceived self-efficacy. *Educational and Psychological Measurement*, *61*(3), 475–489.

Costa, C. J., Aparicio, M., & Cordeiro, C. (2012). A solution to support student learning of programming. *ACM International Conference Proceeding Series*, 25–29. https://doi.org/10.1145/2316936.2316942

Gomes, A. J., Santos, Á. N., & Mendes, A. J. (2012). A study on students' behaviours and attitudes towards learning to program. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 132–137. https://doi.org/10.1145/2325296.2325331

Kumar, R. (2018). *Research methodology: A step-by-step guide for beginners*. Sage.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 14–18. https://doi.org/10.1145/1067445.1067453

M., K. (2014). *Problems in Programming Education and Means of Their Improvement*. 459–470. https://doi.org/10.2507/daaam.scibook.2014.37

Moström, J. E. (2011). *A study of Student Problems in Learning to Program*. http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-48216%5Cnhttp://umu.diva-portal.org/smash/get/diva2:447104/FULLTEXT02%5Cnhttp://umu.diva-portal.org/smash/record.jsf?pid=diva2:447104

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, *39*(4), 204–223. https://doi.org/10.1145/1345375.1345441

Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education*, *18*(1), 1–24. https://doi.org/10.1145/3077618

Siti Rosminah, M. D., & Ahmad Zamzuri, M. A. (2012). Difficulties in learning Programming: Views of students. *1st International Conference on Current Issues in Education (ICCIE2012), October 2014*, 74–78. https://doi.org/10.13140/2.1.1055.7441

Swidan, A., Hermans, F., & Smit, M. (2018). Programming misconceptions for school students. *ICER 2018 - Proceedings of the 2018 ACM Conference on International Computing Education Research*, 151–159. https://doi.org/10.1145/3230977.3230995

Wittie, L., Kurdia, A., & Huggard, M. (2017). Developing a concept inventory for computer science 2. *Proceedings - Frontiers in Education Conference, FIE*, *2017-Octob*, 1–4. https://doi.org/10.1109/FIE.2017.8190459

Xinogalos, S. (2016). Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. *Education and Information Technologies*, *21*(3), 559–588. https://doi.org/10.1007/s10639-014-9341-9

Yusoff, K. M., Ashaari, N. S., Wook, T. S. M. T., & Ali, N. M. (2020). Analysis on the requirements of computational thinking skills to overcome the difficulties in learning programming. *International Journal of Advanced Computer Science and Applications*, *11*(3), 244–253. https://doi.org/10.14569/ijacsa.2020.0110329