INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH in PROGRESSIVE EDUCATION & DEVELOPMENT

OPEN ACCESS JOURNAL

Vol 10, Issue 3, (2021) E-ISSN: 2226-6348

A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review

Rozita Kadar, Naemah Abdul Wahab, Jamal Othman, Maisurah Shamsuddin & Siti Balqis Mahlan
Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Cawangan Pulau Pinang, Pulau Pinang, Malaysia

To Link this Article: http://dx.doi.org/10.6007/IJARPED/v10-i3/11100 DOI:10.6007/IJARPED/v10-i3/11100

Published Online: 21 August 2021

Abstract
Teaching and learning (T&L) computer programming language especially for non-computer science students are challenging. The instructors are facing constraints or obstacles in making the students understand the programming concepts and able to create excellent programming skills to solve the real problem. Lack of logical, creative, and critical thinking among students leads to weaknesses in the implementation of problem-based learning (PBL). The entity of programming language problems such as instructors, students and programming language were examined and analyzed based on related previous studies or research that have been reported. The methodology of Structured Literature Review (SLR) was applied to explore the entities and help to measure the major contributing factors influencing the weaknesses of programming language understanding among students. Hence, SLR can guide researchers to find related information from the articles systematically and extensively in line with the objectives of the research framework.
Keywords: SLR, Instructors, Students, Programming Nature, Programming Problem.

Introduction

Computer programming courses are among the main requirements of study plans in higher education not only in the field of Computer Science and Information Technology, but also in Science, Mathematics, and Engineering field. This is because, in this modern age, knowledge of computer technology and programming in various fields of work is a necessity to fulfil the demand of the industry, which is to have expertise in the field of Information and Communications Technology (ICT) such as software programming, database, software engineering, computer networking and creative multimedia as stated by Rosminah and Zamzuri (2012). Nevertheless, they added that learning computer programming is very complicated and brings a huge challenge to many students.
Writing a computer program is not an easy task. According to Moström (2011), novice students must understand the problem, formulate the solution using standard problem-solving techniques, and write down the solution to solve the problem using a programming language in such a way that a computer can follow the instructions. In reality, learning a programming language is much more complex than this very simplified description since (Baist & Pamungkas, 2017) mentioned that this skill requires other skills from designing the

algorithm, writing program as well as understanding the syntax of programming language. Moreover, (Abdul et al., 2018) mentioned that other than learning the basic concept in language syntax, the students are required to comprehend the language structure and style, able to interpret requirement into an algorithm, translate the program code with the correct syntax, find bugs, apply correct logic and use program development environment.

As stated by (Gomes et al., 2012), several causes that contribute to learning difficulties in computer programming include the students' background knowledge and attitudes, teaching and learning methods, as well as social context. They also added that the lack of problem-solving skills and mathematical background in students as well as programming question proposed by lecturers beyond the students' cognitive development are among the factors of poor performance in programming courses. Another study by (Xinogalos, 2016) pointed out the significance of preparing high-quality instructional materials that are not only based on the teaching and learning design of the course but more importantly, dispels students' difficulties and misconceptions on computer programming. Furthermore, the study suggested that changes are necessary for students' study methods and attitudes as well as the traditional teaching approaches. Instructional methods such as hands-on practice on learning programming and contemporary innovative teaching and learning techniques must be used to increase students' interest in computer education.

Therefore, this paper aims to identify the factors that lead to difficulties and challenges in learning computer programming by novice students and suggest strategies for addressing these issues. It is hoped that this study will assist computer science educators to improve their teaching approaches for basic programming courses, enhance students' interest and increase students' performance in programming subjects.

Methodology

This paper conducted a Systematic Literature Review (SLR) based on the steps proposed by Kitchenham et al. (2009) as in Figure 1.

a) Research Questions

This paper aims to explore the literature review to identify publications related to the issues of the problems that arise in learning programming languages among students.

Several research questions were constructed as stated below:

i. What is the programming nature that becomes a problem for students when learning a programming language as reported by previous work?
ii. What are the problems faced by students when learning a programming language as reported by previous work?
iii. What are the problems faced by educators when teaching programming languages as reported by previous work?

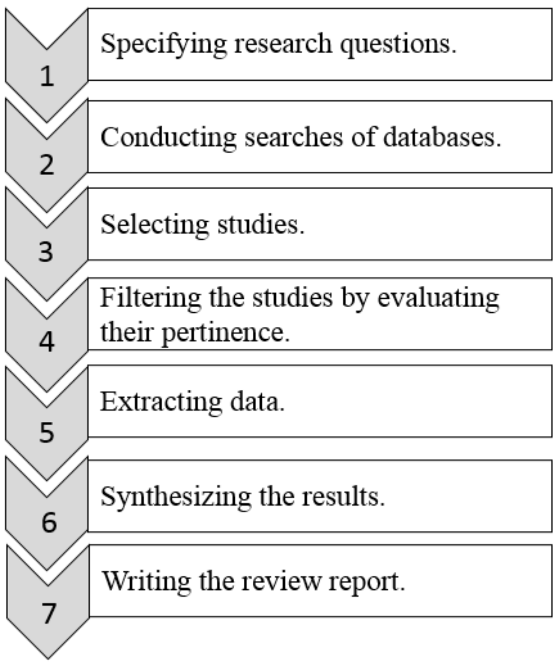
1
Specifying research questions.
2
Conducting searches of databases.
3
Selecting studies.
4
Filtering the studies by evaluating their pertinence.
5
Extracting data.
6
Synthesizing the results.
7
Writing the review report.

Figure 1 Systematic Literature Review (SLR) Process

b) Conducting Searches and Selecting of Databases

Choosing search terms for an inclusive survey of the introductory literature has been proved challenging. Terms that are too general would result in an irregular group of papers, while terms that are too specific tend to lose relevant papers. Thus, the search strategy was designed in such a way that the search process matches the specified. Only quality research papers and answers to specified questions were selected. After some trial and error with various databases, a combined search phrase that seemed to cover the area of interest was chosen. The following table (Table 1) lists the searching strategies implemented while searching databases in this study.

Table 1
Systematic Searching Strategies

Search Guidelines	Searching Strings	Databases
• Search terms are not case-insensitive • Common words are ignored. • AND is implied. • Combine multiple words with OR to find articles containing either terms, such as learn programming OR problem. • Use parentheses to create more complex queries, such as archive ((journal OR conference) NOT theses) • Search for an exact phrase by putting it in quotes, such as "open access publishing". • Use * in a term as a wildcard to match any sequence of characters, such as program* would match documents containing "programming" or "programmer".	"introductory programming" "introduction to programming" "novice programming" "novice programmers" "learn programming" "learning to program" "teach programming" “problem” “programming structure” “programming nature”	• ACM • IEEE Explore • ScienceDirect (Elsevier) • SpringerLink • Scopus

Also, the snowballing process was conducted in both directions (backward and forward). The main goal of this stage was to expand the set of possibly relevant papers by focusing on papers either citing or being cited by each previous selected study. Backward snowballing was proved iterative, as this process was firstly conducted right after the selection process on one depth level, before adding relevant cited papers during the data extraction, thus increasing the depth level.

c) Filtering and Extracting Data
In Figure 2 shows the percentage of selected papers (19%) compared to unselected paper (81%) in the total of 105 extracted papers. Meanwhile, Figure 3 depicts the number of extracted papers by years from 2010 until 2019 (n = 105) and the number of papers filtered (n = 20).

The papers were filtered according to the following criteria:
- Title, keyword list, and abstract explicitly stating that the paper is related to problems in teaching and learning of programming.
- Studies conducting a systematic mapping study or SLR on the problems in teaching and learning of programming.
- Studies providing a state of the art, taxonomy, review on the problems in teaching and learning of programming.
- The paper’s full text is not available for download.
- The publication year starts from 2010.

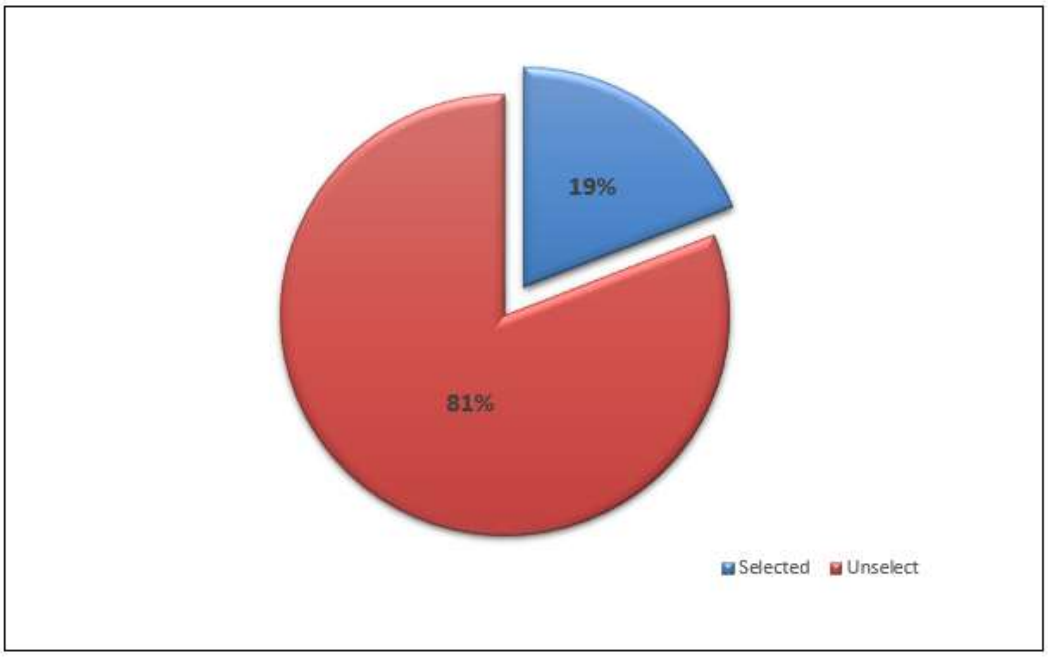

Figure 2 The percentage of selected and unselected papers

Extracted data were organised according to the research question answered. For each study, the publication year and the main topics addressed were recorded followed by reading the title and abstract. Through the extracted software, categories were identified to represent the main problem presented in the studies. They mainly came from previous main topics; if the number of papers related to a specific topic was not high enough, they were not considered as categories, since the aim here is to combine the results from papers dealing with the same subject. Nevertheless, for each category identified, the ideas that emerged from the selected studies were discussed. Other additional relevant information was extracted to develop the discussion.

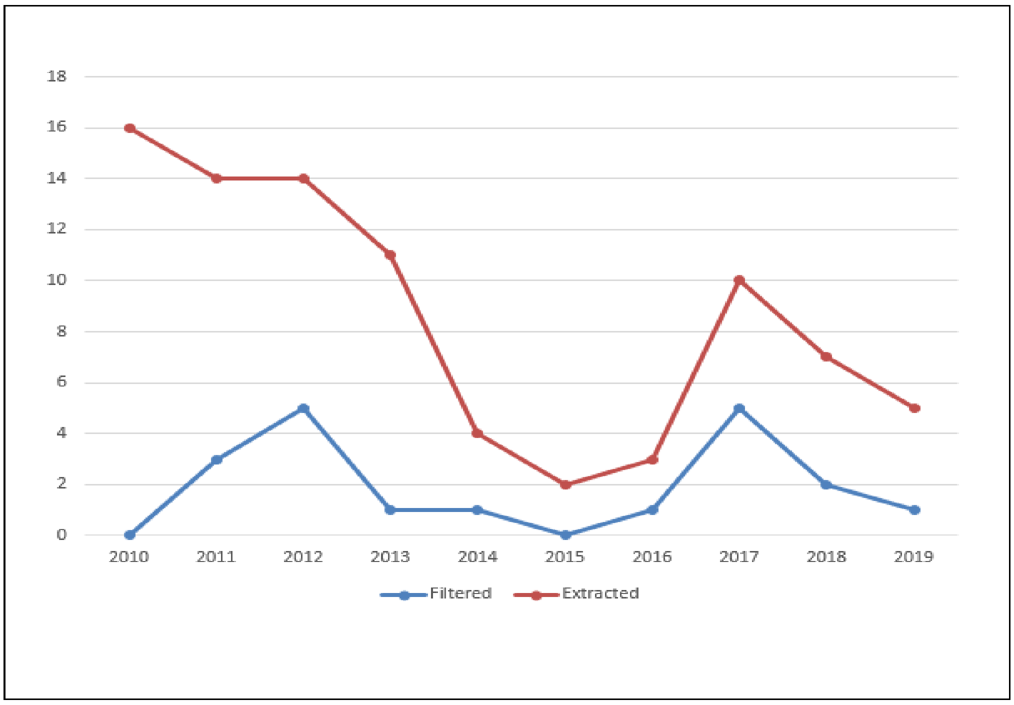

Figure 3 The number of extracted and filtered papers

Result

Based on the previous research on learning computer programming difficulties among students, programming language nature, students and instructors are the three entities that need to be further scrutinised and discussed.

a) Programming Language Nature

The activity in reviewing a program source code is not the same as reviewing ordinary documents and many problems in program comprehension arise due to the use of textual representation as the primary source of information. Programs are often in the form of a hierarchical structure, but the actual behaviour of a program cannot be reflected as it is represented in textual forms. Although many methods and tools have been proposed to represent source code including cross-referencing, developers' program domain knowledge, syntax highlighting and tools, comments, dependence graph, slicing, ripple analysis and program decomposition, experience have shown that textual presentation is the most suitable to represent a program (Krinke, 2004). However, the problems still exist if the source code is used in a form of text-based due to the source code. Thus, this section discusses the issues that arisen in learning a program based on the nature of the programming language.

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. Rosminah and 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 and 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 2.

Table 2
Summary of Problem Exist in Programming Language Natures

Problem	Author(s)	Descriptions
Developing an algorithm	Chan Mow (2008); Lahtinen, Ala-Mutka, and Järvinen (2005); Konecki (2014)	• Lack to combine the syntax with logic and concepts. • Lack of skills in translating problems into a charitable action sequence. • Lack of understanding about good instructional approaches.
Transferring an algorithm to a programming language	Costa, Aparicio, and Cordeiro (2012); Lahtinen, Ala-Mutka, and Järvinen (2005); Xinogalos (2016)	• Unable to transform the problem into programming instruction. Most of the students may know the syntax and semantics of individual statements, but do not know how to combine them into valid programs. • Difficulties in using language libraries such as searching in language libraries, finding the appropriate function, and using it correctly in a program.
Understanding programming structures	Ala-Mutka (2004); Bosse and Gerosa (2017); Chan Mow (2008); Lahtinen, Ala-Mutka, and Järvinen (2005); Qian and Lehman (2017); Swidan, Hermans, and Smit (2018); Wittie, Kurdia, and Huggard (2017); Xinogalos (2016)	• The difficulty of understanding the sequentially of statements, a variable holds value and the interactivity of an input action. • Lack of understanding that each instruction is executed in the state created by the previous instructions. • Difficulties in using notation for representation of a program. Notation refers to the symbols of a programming language and the syntactic rules for combining them into a program. • Pointers, array and data structures are the most difficult programming concepts.

Constructing modularization	Bosse and Gerosa (2017); Xinogalos (2016)	• The difficulty that students have to work with functions. • Lack of understanding of the scope of variables and why it is necessary to pass and return parameters.
Testing and debugging	Bosse and Gerosa (2017; Chan Mow 2008; Pears et al. (2007); Qian and Lehman (2017); Rosminah and Zamzuri (2012)	• Unable to master the use of the compiler, as well as error and warning messages. • Handling syntax error is one of the barriers where the most common problem is the lack of ability to find errors. • Lacking to visualize the program state during code execution. • Mismatched parentheses, brackets, or quotation marks, missing semicolons, and using the illegal start of expressions.

b) Students

Lack of problem-solving skill and limited surface knowledge of programs is among the main factors that lead to difficulties in learning computer programming. As stated by (Abdul et al., 2018), novice students often approach programming line by line instead of using meaningful structured programs. In a study by (Mohamed et al., 2011), they found that novices feel programming is very complicated as it requires too much knowledge and skills to be mastered simultaneously and at an early stage. Therefore, most of them start to write computer program without analysing, designing and understanding the problem thoroughly as mentioned by (Oroma et al., 2012). Many of them do not have the abilities to interpret the problem appropriately since they have a weak approach to problem-solving, thus making most students take the easy way out by plagiarising their friends to complete the assignment given or even worse, (Rosminah & Zamzuri, 2012) said that some of them just expect marks out of sympathy from their programming subject instructors. Their study also discovered that students' difficulties in programming are due to three interrelated issues, which are (1) difficulty in understanding the basic concepts of programming structure, (2) problem in designing a complete task, and (3) syntax of programming languages. They also added that novices are not only lacking the abilities to apply abstract concepts of programming to solve the problem given, but also skill in mastering various processes such as planning, analysing, designing, editing, compiling and debugging program code.

Novice learners also do not understand the programming syntax and constructs. According to (Mohamed et al., 2011), the three main common errors for beginners include syntax errors in basic program structure including semicolons and curly braces, problem in program design or difficulties with basic program structure. Furthermore, in a study by (Qian & Lehman, 2019), students also make syntactic mistakes in their code, such as failing to declare variables, using malformed Boolean expressions, mistakenly using the assignment operator (=) instead of the comparison operator (==), among others. Therefore, due to a lack of experience with programming syntax, (Abdul et al., 2018) said that novices are incapable to read debug error messages given by compiler, thus unable to mend their program bugs appropriately without assistance from their friends or lecturers. Students need to know the basic syntax, structure, and style of a programming language gradually. (Mhashi & Alakeel, 2013) identified several programming structures that are tough for the students to learn, which consist of loops, recursion, arrays, pointers, passing parameters, and abstract data types. Although the theoretical lectures are vital in learning programming, they said that students need practical sessions to understand how to apply these concepts before writing computer programs.

Another factor that influences the difficulties of students in learning programming subjects is poor learning style. Each student has a different learning style. Some of the students prefer group discussion, while others favour independent study. Nevertheless, regardless of any type of learning style, the most important thing is the way students' think. According to (Rahmat et al., 2012), learning to programming involves a different way of thinking. A study by (Oroma et al., 2012) discovered that most people find it easier to learn a certain subject that they are familiar with rather than learning a new subject. This is because the learning process is usually built on previous knowledge as well as experiences, and since computer programming subjects are not related to any former subjects in their primary or secondary schools, many students feel that this new subject is complicated. By thinking this way, (Rahmat et al., 2012) said that most students make a minimal initiative in studying programming subject and depend on assistance from their lecturers, friends or copying computer programs from Internet sources to solve programming assignments without exactly understanding the task given. Novices also rely completely on lecture notes, slides and answer schemes prepared by lecturers although these reference materials are insufficient to increase students' understanding of computer programming.

Finally, a study by (Gomes et al., 2012) revealed the correlation between programming subject grades and student motivation. Due to the negative perception of students on programming, most students put little effort and are demotivated to learn programming with an open mind. This attitude accompanied by weaknesses in problem-solving skills has resulted in low grades in introductory programming courses. For that reason, (Gomes et al., 2012) suggested that programming instructors should observe closely their students' motivation level and try to inspire them through innovative way of teaching styles. This can be done by increasing the number of practical hands-on sessions and two-way interactive tutorial discussion on the basic concepts of programming instead of just delivering the theory through abstract and dull lecture sessions. However, they added that this instructional method is only suitable for a small group of around 20 students or less in a class. With this approach, lecturers can get to know their students more closely and aware of their strengths and weaknesses in computer programming.

c) Instructors

The shortage of professional educators in programming language especially from the industries is quite critical. The lack of qualified professional educators in education institution demotivates the students to see the connection of their studies with future careers (Oroma et al., 2012). Lack of guidance from the instructors especially on the teaching and learning style influence the students' interest in the programming class (Jenkins, 2002). When the students are not being explained clearly of the appropriate learning style, they tend to adopt a common learning style that is inappropriate to the nature of computer programming subjects. The same research also found that the students usually tend to memorise the whole coding whenever they cannot understand the flow of the algorithm.

Some problems are related to the instructor's knowledge and ability to deliver effective instructions to students (Xinogalos, 2016). Poor content knowledge in programming skill leads to difficulties in making the novice students attracted to programming class. The instructors may not be able to see from the perspective of novices' eyes and therefore fail to comprehend students' complications. The inexperience of preparing the pedagogical content knowledge, applying effective teaching strategies and appropriate teaching tools will affect students' misconception' of the subject nature. The supporting content materials prepared by the instructors is also the contributing factors of computer programming problems among students as mentioned by Rahmat et al. (2012). The lecture notes written in English make it difficult for Malay students to understand the contents. The contents in the lecture note that are very brief, examples given that are difficult to understand, exercises that are not provided with the answer schemes for the students to cross-check with their answers and the lecture slides that are taken or copied from the author's slide of the textbook without improvising the slides more comprehensives are among the reasons why the programming class become static or monotonous. Instructors should play a prominent role in delivering the knowledge effectively and explaining the solution of the problems to the students (Gomes & Mendes, 2007).

The lack of sharing relevant examples or real applications by the instructors to the students also contributes to the difficulties of learning computer programming. In addition, the students are not happy with the quality of feedback given by the instructors on the exercises assessed (Mhashi & Alakeel, 2013). The instructors must give enough response and comments on the assessments and demonstrate proper solutions. This will assist the students to understand how the program works. Furthermore, increasing the number of assessments, selecting suitable exercises and illustrating detailed steps of solution will increase the students' understanding and their programming skills. Rosminah and Zamzuri (2012) in their study reported that the students also complained of the hardware and software equipment in the computer laboratories that are not well equipped. The students have to buy the computer peripherals and rent online software for a certain period to accomplish the assessment but ended up with the poor quality of assessment submitted or have a lot of plagiarism works.

Discussion and Recommendations

a) Students' Perception on Programming

It takes years for a novice learner to become an expert programmer. The process of learning programming requires its learners to master multi-layered skills in a short amount of time. Due to this factor, students face difficulties in learning programming not only because of their limited problem-solving skills and having surface knowledge but also their inability to visualise the execution process of the program code. For that reason, they cannot figure out the problem when their program does not turn out the way they want. One of the ways to solve this issue is by using effective learners' assistance such as visualisation tools. These teaching aids can help students to learn programming and understand the important elements related to programming behaviour during program execution. Moreover, visualisation tools have the potential to create a suitable learning environment to help novice learners construct a practical mental model of programming concepts to enhance their understanding.

There is an English proverb saying that practice makes perfect. Therefore, practical or laboratory activities and regular constructive discussions with lecturers or friends can help students to learn fundamental programming effectively. Learning programming needs to involve practical activities and intensive problem-solving exercises owing to its dynamic concept. Hands-on programming sessions in the computer lab will help the students to address the understanding of difficult terminology in programming as well as stimulate students' interest in the field of programming through tutorials carried out. However, instructors are advised to create problem-solving questions related to the students’ field or real life.

Most students have a negative perception about the difficulty of programming subjects from their seniors, which makes them lose interest and have less motivation even before learning it. This attitude has also made them less diligent in the class, leading to low grades in this subject. Because of that, programming instructors need to wisely find ways to increase the level of students’ motivation and remove their negative perceptions of this subject with a lecture delivery style that attracts the attention of students during their first meeting. The classes should be in small groups of less than 30 students so that the instructors can easily recognise the weaknesses of their students. Besides, lecturers should be approachable for students to ask their problems and misconceptions as well as always encourage a two-way discussion during programming class sessions.

b) Real Programming Environment

Representing source code in a form of text-based causes confusion among students because basically, the source is in a structured form. Thus, students need a technique that can help them visualising the structure of source code, which needs to be given attention. For example, to improve students' understanding of programs and data flow, visualisation techniques should be introduced in the process of understanding programming. Other than that, in improving the understanding of program structure, several things should be given attention, such as:

- Increasing the levels of granularity within the source code.
- Representing the multiple levels of abstraction or views in source code.
- Representing the structural information.
- Increasing the ability to view the relationship of concepts.

Another factor is the students' ability to deal with source code errors. Dealing with code errors is one of the important parts of learning programming. Thus, students are encouraged to learn to read the code errors and learn from them as they contain important information that tells them what and where something is wrong. Also, from the errors, defects or unexpected things in program code during the development process, the best way to advance is through debugging. Learning how to debug is an important skill that is very helpful for finding errors and bugs.

c) Educators’ Spirits on Teaching

An instructor is considered as a cover of the book. If the cover looks unattractive, fewer buyers are interested to purchase the book. The instructors must portray outstanding image, appearance, and presentation as well as delivering creativity in teaching the computer programming courses. Excellence in teaching delivery will give a high impact on students understanding and remembrance. The instructors should stress the students' ability to solve the real problem as the ultimate objective to produce excellent graduates. The students should know how to relate and apply appropriate theories and methodologies in their solutions. The roles of instructors are very important to guide the students and slowly train them to work independently; nevertheless, continuous supervision is needed to ensure that the students are on the right track.

The instructors are encouraged to create innovative courseware as teaching aid tools. Nowadays, there are a lot of software and e-learning tools available on the internet or mobile applications. These applications can be downloaded, modified and customised according to the nature of the course or subject. The instructors are free to add any features or contents to make the courseware more amazing and magnificent. For the programming course, the interactive element between students and the instructor is compulsory as it can create an effective learning curve among students and can be integrated with the group activity element to impose active communication.

The instructors should not enforce the students on the syntaxes of the programming language, but on the problem solving before bringing the students on the technicality of the programming language. Nowadays, in most universities, especially in programming subjects, the students are exposed to problem-solving using the tools such as program specification, flow chart or pseudo code before the detailed steps of coding are written. This approach will help and guide the students to produce precise and brilliant solutions to the problem statements. Designing the curriculum of programming subject should start with easier or educational programming languages such as ELISA, Basic or Python programming language before proceeding to a higher level of programming languages such as C, C++ or JAVA. Once the students understood the semantics of the programming language in their earlier class, they will not face any problems learning new programming languages in the following advanced programming classes.

Conclusion

In summary, students are the leading entity that contributes to the lack of computer programming skills. Creating innovative teaching methods and shifting the lecturer-centred learning to students-centred will enhance the students' skills in programming. The instructors should change the teaching paradigm from the conventional face-to-face teaching method to an interactive online teaching method, which is more attractive and effective to boost up the learning curve of programming skills among students.

The problem-based learning (PBL) approach is among the relevant methods to sharpen the skills of solving real problems. Exposing the students to the real problems and proposing the solutions by applying related theories could lead to higher competencies among students, generate a better quality of future programmers and fulfil industrial demands. Future studies are recommended to focus on the best practices among instructors to escalate the programming skills among students using the SLR methods. Varieties of interesting teaching method using attractive tools, platforms and creativity will be the main focus for the e-Learning approach to enrich the knowledge in programming.

Corresponding Author
Naemah Abdul Wahab
Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Cawangan Pulau Pinang, Malaysia
Email: naema586@uitm.edu.my

References

Abdul, T. F. B., Anuar, N., & Said, M. R. F. (2018). How the nature of programming and learning materials affects novice learner's motivation and programming ability. *ACM International Conference Proceeding Series*, 124–128. https://doi.org/10.1145/3178158.3178184

Baist, A., & Pamungkas, A. S. (2017). Analysis of Student Difficulties in Computer Programming. *VOLT : Jurnal Ilmiah 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

Mow, C. 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

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

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. *Information and Software Technology*, *51*(1), 7–15.

Krinke, J. (2004). Visualization of program dependence and slices. *20th IEEE International Conference on Software Maintenance, 2004. Proceedings.*, 168–177. https://doi.org/10.1109/ICSM.2004.1357801

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

Mhashi, M. M., & Alakeel, A. L. I. M. (2013). Difficulties Facing Students in Learning Computer Programming Skills at Tabuk University. *Recent Advances in Modern Educational Technologies*, 15–24.

Mohamed, S., Hamilton, M., & D'Souza, D. (2011). Understanding novice programmer difficulties via guided learning. *ITiCSE'11 - Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science*, 213–217. https://doi.org/10.1145/1999747.1999808

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

Oroma, J., Wanga, H., & Ngumbuke, F. (2012). Challenges of Teaching and Learning Computer Programming in a Developing Country: Lessons From Tanzania. *INTED2012 Proceedings*, *October*, 3820–3826. https://doi.org/10.13140/2.1.3836.6407

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

Qian, Y., & Lehman, J. D. (2019). Using Targeted Feedback to Address Common Student Misconceptions in Introductory Programming: A Data-Driven Approach. *SAGE Open*, *9*(4). https://doi.org/10.1177/2158244019885136

Rahmat, M., Shahrani, S., Latih, R., Yatim, N. F. M., Zainal, N. F. A., & Rahman, R. A. (2012). Major Problems in Basic Programming that Influence Student Performance. *Procedia - Social and Behavioral Sciences*, *59*, 287–296. https://doi.org/10.1016/j.sbspro.2012.09.277

Rosminah, M. D., 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