Journal of Computing Research and Innovation (JCRINN) Vol. 6 No. 3 (2021) (pp13-21)
https://jcrinn.com : eISSN: 2600-8793

Students' Assessments in Learning Programming based on Bloom's Taxonomy

Rozita Kadar[1*], Syarifah Adilah Mohamed Yusoff[2], Saiful Nizam Warris[3], Mohd Saifulnizam Abu Bakar[4]
[1,2,3,4]Faculty Computer and Mathematical Sciences,
Universiti Teknologi MARA, Cawangan Pulau Pinang, Malaysia

*Corresponding author: * rozita231@uitm.edu.my*
Received Date: 28 July 2021
Accepted Date: 26 August 2021
Published Date: 13 September 2021

HIGHLIGHTS

- Problems faced by students while learning a programming language.
- Parameters to measure the level of understanding of students during the learning process.
- Structure of programming at the beginner level that educators need to focus on.
- Forms of questions that can be used to measure students' understanding of programming.

ABSTRACT

Learning a program is important for all students, not only students from the field of computer science but all fields. Programming languages are different from human communication languages as they have different structural forms. This makes it difficult for beginners especially for non-computer science students to understand the structure of programming languages. Therefore, to learn and understand the programming language more effectively, this article focuses on the important structure in learning a program from the initial stage to the advanced level suitable for non-computer science students. The objective of this article is to suggest important elements that can be assessed on these students which are to measure their understanding as they learn programming languages. The questions proposed to measure students' understanding were based on Bloom's Taxonomy, which covers six levels of understanding. It is hoped that this assessment proposal can act as a guideline for educators in fully focusing on important matters during the teaching and learning process.

Keywords: *programming assessment, Bloom Taxonomy, computer education, program structure*

INTRODUCTION

Programming skills are among the strategic planning in Education 5.0. Introduction to programming language is a course offered in several programs in tertiary education. The course emphasizes the introduction of computer problem solving with the use of programming language to illustrate the solution. It aims to develop students' problem-solving strategies, techniques and analytical skills that can be applied to the computer field or in other areas as well as in their subsequent course work and professional development. Students learn to identify a problem, design appropriate solutions, and solve the problem in a computerized way.

In the beginning, students are introduced to the basic elements of programming. At this stage, students are expected to declare variables, apply mathematical operations, and construct a simple programming statement. Then, students are exposed to the use of selection control structure and repetition control structure. The implementation of functions and array are also discussed in this course. At the end of the semester, students should be able to write complete programming by applying all the elements discussed in the course.

Therefore, this paper aims to identify the assessments' terms that led to assess non-computer science students in learning computer programming and suggest forms of questions as an assessment of students in measuring their understanding in learning a programming language. The proposed form of the questions was based on Bloom's Taxonomy. 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.

Related Works

Source code is a more trusted source of data compared to composed documentation primarily since documentation is regularly non-existed or obsolete (Maalej et al., 2014). However, the problems still exist if the source code is used as a reference to a system. The activity in reviewing and understanding a source code is different from 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 found in the form of a hierarchical structure, but the actual behavior of a program cannot be reflected as it is represented in textual forms.

The understanding of a program is "the process of taking source code and understanding it" (Deimel & Naveda, 1990) or the process of using existing knowledge to acquire new knowledge (Aljunid et al., 2012; Xia et al., 2017). It is related to execution behavior and the relationship of variables involved in the program (Hu et al., 2018; Sasirekha & Hemalatha, 2011). The study of program comprehension can be explained as the process that occurs in the learners' minds when they understand a program (Feigenspan & Siegmund, 2012). Table 1 shows a summary of previous studies on the problems that exist among students, especially novices when they learn a programming language.

Table 1: Summary of Problem Exist in Learning a Programming Language

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); Siti Rosminah and Ahmad 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.

Construction of Questions based on Bloom's Taxonomy

One way to measure students' skills in the mastery of programming is through assessments. This article proposes the role of Bloom's Taxonomy in the preparation of student assessments. Bloom's Taxonomy was created by Benjamin Bloom (1956), which proposed six different levels of understanding, namely remember, understand, apply, analyse, evaluate, and create. The six levels of understanding proposed by Benjamin Bloom and the summary of Bloom's Taxonomy explored by Thompson et al. (2008) as the aims of assessments is discussed in Table 2 below:

Table 2: The aims of assessments based on Bloom's Taxonomy

Levels of Understanding	Aims
a) Remember	- Retrieving relevant knowledge from long-term memory - Includes recognising and recalling.
b) Understand	- Constructing meaning from instructional messages, including oral, written, and graphical communications. - Includes interpreting, exemplifying, classifying, summarising, inferring, comparing, and explaining.
c) Apply	- Carrying out or using a procedure in a given situation. - includes executing and implementing.

d) Analyse	- breaking material into its constituent parts and determining how the parts relate to one another and to an overall structure or purpose. - Includes differentiating, organising, and attributing.
e) Evaluate	- Making judgements based on criteria and standards. - Includes checking and critiquing.
f) Create	- Putting elements together to form a coherent or functional whole, reorganising element into a new pattern or structure. - Includes generating, planning, and producing.

Suggested Questions based on Bloom's Taxonomy for Non-computer Science Student

As non-computer science students, there are some important things that they need to learn in programming. This section proposes the features of programming language that need to be mastered by the students while learning programming, which are divided into sections: introduction to programming; basic elements of programming; selection and repetition control structures; function; Introduction to programming exposes students to the concept of programming where in this section, students need to master the terms of programming, know about the benefits and importance of programming in life.

Students also need to be exposed to the evolution as well as the history of the development of programming languages. Another important thing is that the students need to be exposed to the models that can be used during the development of a system like PDLC. With these, students can make analyses and design solutions in the best way. From these sections, this article proposed the questions that should be assessed on students. Bloom's Taxonomy was used as a guideline in preparing the questions. The suggested assessment terms are listed in Table 3 below:

Table 3: The Suggested Assessments' Terms based on Bloom's Taxonomy

Levels of Understanding	Assessment Terms
a) Remember	• Identifying a particular construct in a piece of code. • Recognising the implementation of a subject area concept. • Recognising the appropriate description for a subject area concept or terms. • Recalling any material explicitly covered in the teaching programme. eg. conceptual definition.
b) Understand	• Translating an algorithm from one form of • representation to another form. • Explaining a concept or an algorithm. • Presenting an example of concept or an algorithm.
c) Apply	• The process and algorithm are known to the learner, and both are applied to a problem that is familiar, but that has not been solved previously in the same context. • The process and algorithm are known to the learner, and both are applied to an unfamiliar problem.
d) Analyse	• Breaking a programming task into its component • parts (classes, components, etc.). • Organising component parts to achieve an overall objective. • Identifying critical components of a development.

	Identifying unimportant components or requirements.
e) Evaluate	• Determining whether a piece of code satisfies the requirements through defining an appropriate testing strategy. • Critiquing the quality of a piece of code based on coding standards or performance criteria.
f) Create	• Coming up with a new alternative algorithm or hypothesising that a new combination of algorithms will solve a problem. • Devising an alternative process or strategy for solving a problem; or complex programming tasks, this might include dividing the task into smaller chunks to which they can apply known algorithms and processes. • Constructing a code segment or program either from an invented algorithm or through the application of known algorithms in a combination that is new to the students.

In the next section discusses about the structures of programming at the beginner level for non-computer science students that educators need to focus on and the forms of questions that can be used to measure students' understanding of programming. The suggested structures area based on references from Malik (2011), Cay and Timothy (2013), Bjarne (2014) & Niccolai (2012).

Introduction to Programming

Introduction to programming exposes students to the concept of programming where they need to master the terms of programming, know about the benefits and importance of programming in life. Students also need to be exposed to the evolution as well as the history of the development of programming languages. Students should also be exposed to the models that can be used during the development of a system like Program Development Life Cycle (PDLC). With that, students can make analyses and design solutions in the best way. The following are suggested questions used to test students' familiarity with the introduction to programming:

- Define/explain the terms: program/programming, source code/ source file, integer, floating-point, object code, assembler, compiler, interpreter, types of the programming design approach, types of error, algorithm.
- Briefly explain the importance/advantages of programming.
- Identify the difference/distinguish between machine language and high-level language, assembler, compiler and interpreter.
- Write/insert comments in a program; how the compiler responds to the comments.
- Identify/briefly explain the steps in Program Development Life Cycle (PDLC).
- Illustrate/write pseudocode or draw a flowchart based on a problem situation and vice versa.
- Write/insert indentation in a given program segment.

Basic Element of Programming

The things that need to be revealed to students are the basic elements needed when designing a program. The main thing that students need to know is the programming structure of a language. Different programming languages take different forms. Thus, students need to be proficient in using some important elements when structuring programming. The role of variables and data types is very important for students to know. Writing declaration statements, input/output statements, assignment statements are the main things that require students' skills before writing programming.

Students should also know how to use arithmetic expressions and mathematical symbols. In addition, dealing with errors in the program should also be emphasized to students. Therefore, to test students' understanding of this matter, the following questions were suggested:

- Identify the rules of naming identifier and determine the validity of identifier, how an identifier is related to reserve word.
- Write declaration statements and assignment statements for variables and constants to accomplish the given sentences.
- Evaluate/trace the mathematical expression based on certain input – basic operations (+.-./.*.%), function pow(), sqrt(), etc.
- Postfix increment and prefix decrement - a++, a--, ++a, --a., unary and binary operators, compound assignment.
- Give a problem situation and transform it into a C++ program segment.
- Give a program and find/label syntax errors.
- Convert algebraic equations into a programming statement.
- Construct/evaluate the Boolean expression using relational & logical operators.
- Write a formatting statement to display decimal numbers.

Control Structure: Selection and Repetition

The advanced thing in programming is to use the control structure in the program, that is, Selection Control Structure and Repetition Control Structure. In understanding this structure, students need to know the role of these two structures, the structure itself and the type of structure. It is pertinent to consider several important elements in writing selection and repetition statements, which require the students' skills in tracing these two structures. The following are the suggested question forms for these structures:

Selection Control Structure

- Discuss the types of selection control structure
- Construct a Selection statement based on the given problem situation.
- Trace program segment and show the output based on the given input.
- Write a complete program that includes the combination of selection & repetition control structure.

Repetition Control Structure

- Discuss the types of repetition control structure
- Identify Loop Control Variable (LCV), starting value, loop condition, updating condition.
- The difference/distinguishment between loop structures
- Trace a program segment and show the output based on the given input.
- Write a program segment based on the given problem situation.
- Rewrite using another looping structure based on the given loop.
- Find the number of loops.

- How to avoid infinite loop from the given program segment.
- Write a complete program that includes the combination of selection and repetition control structure.

Function and Array

A function is a set of statements used to perform a specific task, whereas an array is a data structure that can store a collection of fixed elements of the same data type. Both structures will make the program more efficient in system development. Students need years of understanding of the role of these two structures. There are two types of functions in a programming language; thus, students need to be proficient in using both types of functions.

The important thing in Function is the user-defined function, in which when in writing a function, some important elements need to be understood. Students also need to understand how to trace functions. In handling arrays, students need to be proficient in creating arrays; that is, they need to consider the use of variables and data types for initializing arrays as well as accessing array elements. The following are suggested forms of questions related to the use of functions and arrays:

Function

- Discuss the types of function and its elements: function prototype, function call, function definition.
- User-defined function-strcpy(), strlen(), etc.
- Identify the types of parameters: actual and formal
- Identify the types of variables: local and global
- Identify the types of parameters passing: by values and by references
- Write a function definition based on the given problem situation including receives and return values.
- Trace a program segment including receives and return value; then, show the output.
- Write a complete program by constructing functions including the combination of selection and repetition control structure.

Array

- Write a declaration and assignment statement to an array.
- Write a program segment to input and display data in an array.
- Find the last index number or value in the array.
- Trace program segment and show the output based on the given input.
- Write a complete program by constructing arrays including the combination of function, selection and repetition control structure.

CONCLUSION

This work has discussed the role of Bloom's Taxonomy as a guideline in the preparation of assessments in the context of learning a program. It is very important to look thoroughly at the cognitive processes while preparing the assessments and needs to be taken seriously so that students' ability to master the programming knowledge can be measured more efficiently. This study emphasizes the importance of structure in learning a programming language, as well as the areas where students' understanding should be tested. Bloom's Taxonomy's six levels of comprehension are used to design the questions. As a result, we can determine the objectives for each stage. We presented the assessment term and came up with the proposal questions based on the goals. The proposed questions are organised into sections depending on the programming structures that students would encounter while learning a program. These suggested questions are aimed at evaluating the abilities of students who non-computer science field. With this study, it is hoped that educators will benefit from it by generating more discussion with quality formative and summative assessments that can meet the current needs.

REFERENCES

Aljunid, S. A., Zin, A. M., & Shukur, Z. (2012). A Study on the Program Comprehension and Debugging Processes of Novice Programmers. *Journal of Software Engineering*, *6*(1).

Bjarne, S, *Programming: Principles and Practice Using C++*, 2nd, Amazon Ltd, 2014, ISBN: 13: 978-03219 3.

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

Bloom, B. S. (1956). *Taxonomy of educational objectives*. Vol. 1: Cognitive domain. New York: McKay, 20-24.

Cay S. H. ,Timothy A. *Budd, Big C++ paperback*, 3nd, Wiley, 2013, ISBN: 9781118674291 1.

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

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

Deimel, L. E., & Naveda, J. F. (1990). Reading Computer Programs: Instructor's Guide to Exercises. In *Educational Materials CMU/SEI-90-EM-3* (Issue August). DTIC Document.

Feigenspan, J., & Siegmund, N. (2012). Supporting comprehension experiments with human subjects. *2012 20th IEEE International Conference on Program Comprehension (ICPC)*, *6*, 244–246. https://doi.org/10.1109/ICPC.2012.6240494

Hu, X., Li, G., Xia, X., Lo, D., & Jin, Z. (2018). Deep code comment generation. *Proceedings of the 26th Conference on Program Comprehension*, 200–210.

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

Maalej, W., Tiarks, R., Roehm, T., & Koschke, R. (2014). On the Comprehension of Program Comprehension. *ACM Transactions on Software Engineering and Methodology*, *23*(4), 1–37. https://doi.org/10.1145/2622669

Malik, D.S, *C++ Programming: From problem Analysis to program Design*, 8th Edition, Cengage Learning. 2011, ISBN: 1337102083

Niccolai M. J. , *The C++ Standard Library: A Tutorial and Reference*, 2nd, Addison Wesley, 2012, ISBN: 13: 978-03216

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

Resnick, M.(2019, Jan 3). The Next Generation of Scratch Teaches More Than Coding. *EdSurge*. Retrieved from https://www.edsurge.com/news/2019-01-03-mitch-resnick-the-next-generation-of-scratch-teaches-more-than-coding.

Sasirekha, N., & Hemalatha, M. (2011). Program Slicing Techniques and its Applications. *International Journal of Software Engineering & Applications*, *2*(3), 50–64.

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

Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008, January). *Bloom's taxonomy for CS assessment*. In Proceedings of the tenth conference on Australasian computing education-Volume 78 (pp. 155-161).

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

Xia, X., Bao, L., Lo, D., Xing, Z., Hassan, A. E., & Li, S. (2017). Measuring program comprehension: A large-scale field study with professionals. *IEEE Transactions on Software Engineering*, *44*(10), 951–976.

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