Student Misconceptions about Loops in Introductory Programming Courses and the Influence of Representations

Dimitri Eckert
Engineering Education Research Group
Hamburg University of Technology
Hamburg, Germany
dimitri.eckert@tuhh.de

Dion Timmermann
Braunschweig, Germany
https://orcid.org/0000-0003-0859-1450

Christian Kautz
Engineering Education Research Group
Hamburg University of Technology
Hamburg, Germany
kautz@tuhh.de

***Abstract*—Research Work in Progress Paper: Loops are a fundamental concept of programming. For novice programmers loops generally and nested loops specifically pose big difficulties. This paper identifies common student misconceptions about loops in a first-year C++ introductory computer science course and the relation of these misconceptions to representations. These findings could help design effective instructional methods.**

***Index Terms*—misconceptions, model-eliciting activities, computer science, first year, process modeling**

I. Introduction & Literature

There are many misconceptions held by novice programmers that hinder the fast and thorough development of coding and computational thinking skills. Our goal is to study how external representations can be used to help novice programmers overcome existing misconceptions about loops or even prevent them from occurring. The first step was to gather known misconceptions about loops from literature and verify that they also occur in the observed setting at the introductory computer science course for engineers at our university. The results of this research are presented in this paper. The next step is the analysis of existing internal and external representations used by students and their relationship with discovered misconceptions. Preliminary findings are presented in this paper.

A. Misconceptions about loops in programming

In this paper the term misconception is used to mean students' "understandings or explanations that differ from what is known to be scientifically correct" [1] .

There have been various efforts to identify misconceptions and typical student difficulties, prevent their occurrence and mitigate their effect on the learning success of students. Several authors have catalogued these misconceptions and difficulties documented in literature [2]–[4]. Key difficulties - most of which are misconceptions - related to loops are:

1) General
   a) Belief that the loop body is always executed just once [5].
   b) Grouping of the statements inside the loop body: When there are two statements in the loop body, the students assume that the first statement is executed as often as specified by the loop and only then the second is executed as often as specified [5].
   c) Expectation that a loop produces the exact same output for each iteration and not accounting for changing variables that can also change the output for different iterations [5].
   d) Confusion about what part of the code belongs to the loop body and what does not [6], [7].
2) Control variables and termination condition
   a) Belief that for-loop control variables do not have values inside the loop or that their values can be arbitrarily changed [6], [7].
   b) Struggle to identify the correct range of the control variables of loops [5].
   c) Difficulties in understanding how automated changes to for-loop control variables happen [8].
   d) False conception that a while-loop terminates as soon as the termination condition changes to false and not when the termination condition is evaluated and not met [8], [9].
   e) Misconception that a for-loop control variable constrains the values that can be read from input within the loop [6], [7].
3) Nested Loops
   a) Reference [10] has found that some students assume that nested loops are executed simultaneously as if they were one loop with two control variables and two termination conditions that are changed and checked simultaneously.
   b) Misconception that two nested loops act like two loops that are executed sequentially [11], [12].

B. Methods for identification of misconceptions in programming

Various methods have been used to detect misconceptions – quantitative as well as qualitative methods. Quantitative methods usually consist of written assessments conducted with

a large number of participants. The type of tasks in these assessments varies a lot. Some tasks ask students to predict the output of a given code segment [5], [11], [12]. Some tasks ask students to debug given code [11]. Sometimes students have to write programs to solve a specific problem [10], [11], [13]. Sometimes students are asked to describe the behavior of given code snippets [5], [10].

Qualitative methods are for example executed by semi-structured think-aloud interviews where students solve programming problems while simultaneously explaining what they are thinking [14]. These can lead to a deeper insight into how students solve problems and allows to spontaneously ask further questions where needed. The problems can be as diverse as with quantitative methods. Many researchers use a mixed methods approach combining qualitative and quantitative methods [5], [10], [11].

C. Multiple external representations in programming

Representations are referred to as "a range of transformations that conceptualize, visualize, or materialize an entity into another format or mode" [15]. These representations can be internal or external. Multiple external representations (MER) are a set of external representations about the same item.

Reference [16] has elaborated on the three main benefits of MERs in the context of education: Complementing information or processes, constraining interpretation of one representation by the other(s), and constructing deeper understanding. Especially the latter two could be of use when teaching programming concepts.

The following examples show the possible benefits of MER in programming:

A successful application of MER in a programming environment has been implemented in the form of a development environment that uses multiple representations for the probabilistic programming language Infer.NET [17]. The study conducted with undergraduate students has shown clear benefits such as a reduction of the time, keystrokes, and deletions needed to complete programming tasks.

In [18] students' ability to switch between different external representations is investigated. In this research three representations were analyzed: the task instructions, the coding environment and the actual robot. It was found that the ability to switch between these representations is elementary for the successful completion of the task.

In the next section we will elaborate on the methodology and methods used for the identification of misconceptions in this study. The third section elaborates on the results of these investigations. The fourth and final section of this paper summarizes the findings and mentions necessary future work.

II. Methods

A. Grounded theory

We employed grounded theory in this research as it is suitable for a preliminary, exploratory study as this one. It is "a general methodology with systematic guidelines for gathering and analyzing data to generate middle-range theory" [19]. According to this theory researchers simultaneously collect data and analyze it. Continuously, data is collected, coded into categories, the codes and the data compared again and further analyzed to generate new leads.

B. Context

The study reported here was conducted in two lectures of a first semester introductory computer science course for non-majors from various study programs with voluntary attendance. For the interviews students of this course were asked to voluntarily participate. This course is suitable for this study, as it has a large number of students with varying backgrounds and from varying study programs and covers basic concepts of programming. 590 students were officially enrolled in the course. A computer science major course of this size does not exist at our university and as this study covers basic concepts at a beginners level the results are likely to be very similar for computer science majors. Prior to the start of the semester a preparatory programming workshop was offered. This workshop lasted one day and participation was optional. It covered basic programming concepts including loops.

C. Tests

This paper tries to investigate what misconceptions and difficulties about loops are represented in the context of a first year introductory computer science course at our university (and how they match the ones found in literature) and how they influence the students way of thinking. To answer the former quantitative question written tests were used. For the latter question which is of qualitative nature semi-structured interviews were used. Thus, overall this study employed a mixed methods approach.

1) Written tests: In order to assess the student's understanding of the concept of a loop we created a written pretest containing three test items. All three items consist of three parts. The first part presented the students with a small piece of code and asked them to write down the corresponding output. In the second part they were asked to explain in words what was happening during the execution of the code. In the third part the students had to state how sure they were about the answer they had given.

The code of the first item contained a single loop. The code of the second item contained two nested loops with independent control variables and termination conditions. In the third item it contained two nested loops where the termination condition of the inner loop was dependent on the control variable of the outer loop.

Additionally, we asked the students about their programming experience prior to the course, their study program, if they had attended the preparatory programming course before the start of the semester and if this is their first study program. They were also asked to provide a self-generated identification code to allow us to match their pre- and posttest in an anonymous way [20].

The pretest was conducted during a face-to-face lecture in the 6th week of the semester. It was intentionally conducted

after basic programming concepts such as loops had already been introduced, as otherwise the students would not have been able to complete the test in a meaningful way. The basic concept of loops is still very new at this stage and it is possible that the continuous use of loops during the semester helps overcome initial difficulties. Out of the 590 enrolled students only about 200 were attending the lecture when the test was conducted which may be because attendance is optional. 190 participated in the pretest. The students were given roughly 15 minutes to complete the test.

The posttest was conducted in the last – 14th – week of the semester. Due to the aggravating CoViD-19 pandemic situation during the semester the lecture was held online, therefore the posttest was conducted online as well. 62 students participated in the posttest. The lower number of participants compared to the pretest was mainly due to the decreasing rate of attendance of the course, as only around 70 were still attending the course by that time.

Apart from an additional fourth item and other minor changes the structure of the posttest was the same as that of the pretest. The additional item asked the students to define the major difference in the code structure between the second and the third item. The code snippets were different from the ones in the pretest but had the same overall structure and level of complexity, with one exception: There is no instance where the inner loop in the third code snippet is not executed at all. This had an unintended effect on the discovered misconceptions as discussed in the results below.

2) Interviews: In order to better understand the ways in which students solve the programming problems described in the previous section and how they use representations to explain the discussed problems, semi-structured think-aloud interviews were conducted with 7 students from this course. Due to the CoViD-19 pandemic situation the interviews were conducted via videoconference. Each interview lasted approximately one hour.

The interview contained all the questions included in the posttest as described above with different pieces of code of the same structure and level of complexity. In order to cover corner cases for each of the three items variations of the code snippets were presented to the students with the request to describe how the behavior and the output of the code changed. Additionally, the students were asked to describe the code in the second test item using an image, a metaphor, a diagram of their own choosing and or a flow chart.

III. Results and Discussion

A. Written test

We analyzed the answers of the students concerning the output of the code snippets. For this we used constant comparative method by comparing different wrong answers and creating codes where the pattern suggests the same reason for the answers. Incorrect answers have been coded into 22 categories of errors, while one answer can be assigned to multiple categories. There was also a significant number (87 students (46%) in the pretest, 24 (39%) in the posttest) of incorrect answers that we could not categorize. Among these 22 categories 12 were concerned with loops. Out of these, four were by far the most frequent:

1) Assumption that the control variables of both loops change at the same time: 70 students (37%) in the pretest and 12 students (19%) in the posttest made this mistake. From the output and the textual explanations of the code they had written down we could deduct that the students assumed that two nested loops worked like one loop that contained two control variables that are updated with each iteration of the loop. An example of this can be seen in Fig. 1. This misconception is the same as bullet point 3)a) in the literature review.

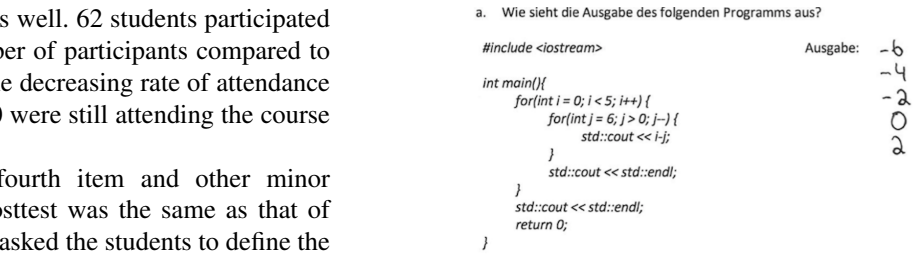

Fig. 1. Example of misconception 1: As can be seen there is only one number per line and the difference between the numbers is 2, which stems from the conception that the two control variables change simultaneously.

2) Confusion about relational operators or misconception of order of execution of conditional statement and loop body: This error occurred in the answers of 65 students (34%) in the pretest and 16 students (26%) in the posttest. Here the students either confused the meaning of the relational operators ($<, >, <=, >=$) or they had a misconception about when which part of the for-loop is executed. Based on the written test, these two explanations could not be distinguished. For a detailed example see Fig. 2. This error is listed as 2)b) in the literature review, but the reference doesn't provide a possible explanation for its occurrence.

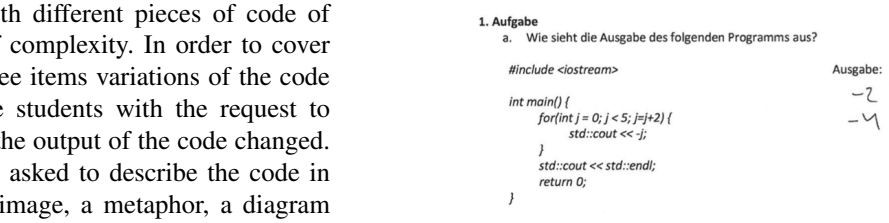

Fig. 2. Example of error/misconception 2: The answer of the student lacks a "0" which possibly stems from a confusion about the range of the control variable or a misconception about at what time it is incremented.

3) Not executing part of outer loop after inner loop, if inner loop is not executed at all: Students had the misconception that if the inner loop of two nested loops is not entered at all for the current iteration of the outer loop (because the condition is not met when it is tested for the first time for the current iteration of the outer loop) then the program skips the remaining part of the body of the outer loop and starts the next iteration of the outer loop. 32 students (17%) made this

mistake in the pretest. An example of this misconception can be seen in Fig. 3. It did not show up in the posttest. This is because in the posttest the inner loop is always executed at least once.

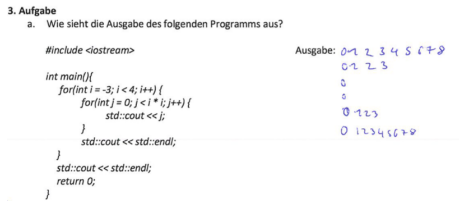

Fig. 3. Example of misconception 3: Between line 3 and 4 of the student's answer an empty line is missing. This stems from the conception that the 8th line of the code is not executed when the inner loop condition is not fulfilled when it is checked for the first time - the first time for this iteration of the outer loop.

4) Confusion about what belongs to the loop body: 39 students (21%) in the pretest and 8 (13%) in the posttest either wrote every number in the output on a new line (in the pretest) or all numbers on the same line (in the posttest) even though the actual output looked different. An example for the pretest can be seen in Fig. 4. We assume this error stems from a misconception about what parts of the code belong to the loop body and what statements do not but it could also just be a careless mistake. This error is the same as bullet point 1)d) in the literature review.

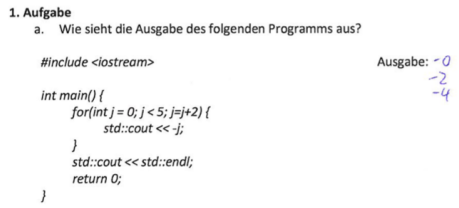

Fig. 4. Example of error/misconception 4: The correct solution shows all numbers on one line and not in a column as in the students answer. A possible reason for this answer is the misconception that the 7th line in the code belongs to the loop body and is executed for every loop iteration.

B. Interviews

All of the 7 students included in the interview sessions were able to solve the problems posed in the interviews correctly apart from some careless mistakes. Therefore, it was not possible to analyze the deeper nature of the misconceptions discovered in the written tests. These students performed significantly better than the average in the written tests. This may be due to a selection bias as students with a higher motivation for the course and a better understanding of the material are maybe more likely to participate in a voluntary interview like this.

Nevertheless, the representations the students were asked to draw allowed insight into their understanding of loops that might explain some of the misconceptions documented in literature and the written test:

1) Linear vs. Nonlinear: Many of the interviewed students drew linear representations that showed the sequential execution of parts of the program. Only few of them were able to include components in their representations that expressed the repetitive aspect of the loops - even after being explicitly prompted to make the representation more compact. This could hint at a lack of abstraction of the loop concept.

2) Explicit vs. Abstract: Most students were representing very explicitly what was happening in the individual steps of the program as opposed to a more abstract form of representation that would also be valid if the parameters of the code would change. Again, this could be an indication of a low-level understanding of loops.

3) Output vs. Variables vs. Processes: Many students focused on the output of the code while constructing the representations. Others were more concerned about the variables and their contents. Only few focused on the processes of the program, such as the checking of the looping condition or the assignment of values to variables. This could be an indication of how students think about the problem and why certain aspects are more difficult for them to grasp.

IV. Conclusion

The conducted research was successful in identifying several misconceptions and difficulties, four of which were held by a considerable number of students. Out of these four one (number 3) had not been discussed in literature beforehand. Additionally our data allow greater insight into difficulty 2. Although the prevalence of the misconceptions and difficulties decreased during the semester, they are persistent for a significant number of students. A limitation of this study is the voluntary basis for the tests which might lead to a selection bias. None of the students in the interviews held these misconceptions and difficulties. Therefore, a deeper analysis of how students came to these misconceptions and difficulties was not possible. Nevertheless, the representations constructed by the interviewed students give some hints at what the students are struggling with. The interviews showed students' difficulties in providing representations that reflected the nonlinearity of the given code and often lacked the expected abstraction and focus on processes. These observations indicate that the interviewed students might have difficulty to think about code in a nonlinear, abstract and process-oriented way. Focussing on multiple external representations might help students overcome these difficulties. Further work needs to be done to analyze the possible connection between the way students represent loops and the discovered misconceptions and difficulties. This could help understand where these misconceptions and difficulties come from and how representations can be used to improve the way loops are being taught.

V. Acknowledgment

We thank Ferdinand Kieckhäfer for his help with the conduction of one interview and giving methodical feedback on it.

REFERENCES

[1] M. J. Leonard, S. T. Kalinowski, and T. C. Andrews, "Misconceptions Yesterday, Today, and Tomorrow," LSE, vol. 13, no. 2, pp. 179–186, Jun. 2014.

[2] L. Chiodini, "A Curated Inventory of Programming Language Misconceptions," in ITiCSE '21: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education, p. 7, Dublin, Ireland, 2021.

[3] Y. Qian and J. Lehman, "Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review," ACM Trans. Comput. Educ., vol. 18, no. 1, pp. 1–24, Dec. 2017.

[4] J. Sorva, "Visual Program Simulation in Introductory Programming Education," Ph.D. dissertation, Dept. of Computer Sc. and Eng., Aalto Univ., Espoo, Finland, 2012.

[5] S. Grover and S. Basu, "Measuring Student Learning in Introductory Block-Based Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic," in Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle Washington USA, pp. 267–272., Mar. 2017.

[6] R. T. Putnam, D. Sleeman, J. A. Baxter, and L. K. Kuspa, "A Summary of Misconceptions of High School BASIC Programmers," Journal of Educational Computing Research, 2(4):459–72., 1986.

[7] D. Sleeman, R. T. Putnam, J. Baxter, and L. Kuspa, "Pascal and High School Students: A Study of Errors," Journal of Educational Computing Research, 2(1):5–23., 1986.

[8] B. du Boulay, "Some Difficulties of Learning to Program," Journal of Educational Computing Research, 2(1):57–73. 1986.

[9] R. D. Pea, "Language-Independent Conceptual 'Bugs' in Novice Programming," Journal of Educational Computing Research, vol. 2, no. 1, pp. 25–36, Feb. 1986.

[10] I. Cetin, "Teaching Loops Concept through Visualization Construction," Informatics in Education, vol. 19, no. 4, pp. 589–609, Dec. 2020.

[11] I. Cetin, "Students' Understanding of Loops and Nested Loops in Computer Programming: An APOS Theory Perspective," Canadian Journal of Science, Mathematics and Technology Education, vol. 15, no. 2, pp. 155–170, Apr. 2015.

[12] M. Mladenović, I. Boljat, and Ž. Žanko, "Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level," Educ Inf Technol, vol. 23, no. 4, pp. 1483–1500, Jul. 2018.

[13] E. Soloway, J. Bonar, and K. Ehrlich, "Cognitive strategies and looping constructs: an empirical study," Commun. ACM, vol. 26, no. 11, pp. 853–860, Nov. 1983.

[14] R. Beichner, "An introduction to physics education research," in Getting Started in PER, vol. 2, 2009.

[15] H.-K. Wu and S. Puntambekar, "Pedagogical Affordances of Multiple External Representations in Scientific Processes," J Sci Educ Technol, vol. 21, no. 6, pp. 754–767, Dec. 2012.

[16] S. Ainsworth, "The functions of multiple representations," Computers & Education, vol. 33, no. 2–3, pp. 131–152, Sep. 1999.

[17] M. I. Gorinova, A. Sarkar, A. F. Blackwell, and D. Syme, "A Live, Multiple-Representation Probabilistic Programming Environment for Novices," in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose California USA, pp. 2533–2537, May 2016.

[18] L. A. Barth-Cohen, S. Jiang, J. Shen, G. Chen, and M. Eltoukhy, "Interpreting and Navigating Multiple Representations for Computational Thinking in a Robotics Programming Environment," Journal for STEM Educ Res, vol. 1, no. 1–2, pp. 119–147, Dec. 2018.

[19] K. Charmaz and L. L. Belgrave, "Grounded Theory," in The Blackwell Encyclopedia of Sociology, G. Ritzer, Ed. Oxford, UK: John Wiley & Sons, Ltd, 2015.

[20] J. Direnga, D. Timmermann, J. Lund, and C. Kautz, "Design and Application of Self-Generated Identification Codes (SGICs) for Matching Longitudinal Data," in SEFI 2017 Proceedings, Tampere, Finland, Sep. 2016, p. 9.