Informatics in Education, 2023, Vol. 00, No. 00, –

DOI: 10.15388/infedu.2023.20
ACCEPTED VERSION (WITHOUT THE JOURNAL LAYOUT).

Understanding loops: What are the misconceptions of lower-secondary pupils?

Jiří Vaníček[1], Václav Dobiáš[1], Václav Šimandl[1]
[1]*Department of Information Science, Faculty of Education, University of South Bohemia in České Budějovice, České Budějovice, Czech Republic*
e-mail: vanicek@pf.jcu.cz, dobias@pf.jcu.cz, simandl@pf.jcu.cz

Received: June, 2022

Abstract. The article describes a study carried out on pupils aged 12-13 with no prior programming experience. The study examined how they learn to use loops with a fixed number of repetitions. Pupils were given a set of programming tasks to solve, without any preparatory or accompanying instruction or explanation, in a block-based visual programming environment. Pupils' programs were analyzed to identify possible misconceptions and factors influencing them. Four misconceptions involving comprehension of the loop concept and repeat command were detected. Some of these misconceptions were found to have an impact on a pupil's need to ask the computer to check the correctness of his/her program. Some of the changes made to tasks had an impact on the frequency of these misconceptions and could be the factors influencing them. Teachers and course book writers will be able to use the results of our research to create an appropriate curriculum. This will enable pupils to acquire and subsequently deal with misconceptions that could prevent the correct understanding of created concepts.

Keywords: Misconception, Learning, Programming, Secondary school, Loop, Repeat, Bebras Challenge, Blockly

1. Introduction

In this article, we leave aside programming as the professional training of an expert and instead consider it – in accordance with Gander (Gander, 2014, p. 7) ¬– as a part of general education. As well as providing an opportunity to explore the world from another point of view, we can see programming as an environment or a microworld, which develops an individual's potential and mental ability, allowing them to acquire new skills (Scherer et al., 2019; Liao, 2000) and understand new concepts and the relations between them. This specification is the basis for an educational aim, which should be strived for. It is also a ground for a choice of suitable educational content, environment, motivation, and methods.
Xia (2017) defines teaching programming as supporting students to understand the concepts of programming via hands-on experience, and learning as the activity of obtaining useful programming knowledge and skills by studying.
Programming is more than just coding. It exposes students to computational thinking which involves problem-solving that uses computer science concepts, such as abstraction and decomposition. Even for non-computing majors, computational thinking

is applicable and useful in daily life (Lye and Koh, 2014). There is evidence of correlations between computational thinking and spatial ability, reasoning ability, and problem-solving ability (Roman-Gonzalez et al., 2017).
However, there is a need for more effective learning environments and learning methods to teach computational thinking (Dagienė and Futschek, 2019).
According to Papert (1980, p. 122), learners can become the active, constructing architects of their own learning. As Ackermann (2010, p. 2) suggests, it is worth offering opportunities for children to engage in hands-on explorations that fuel the constructive process. In line with Bers (2017) and Bers et al. (2019), these views support the idea of programming as a training playground where children construct their knowledge. The question is how to design this playground in the best and most effective way.

2. Background

2.1. Concepts in learning programming

One of the key conditions for acquiring skills is to understand the concepts. The process of active learning of these concepts is described in the theory of generic mental models (Hejný, 1990, p. 58; Hejný, 2012, p. 44) which works with the mechanism of cognitive process. According to this theory, the process of constructing a piece of knowledge is based on the construction of isolated models, resulting in the creation of universal 'generic models' (Hejný, 2012, p. 44). With enough time and the opportunity to experience a sufficient number of isolated models in different situations, one is able to build a generic model which should work in all known situations (Hejný, 1990, p. 59).
Basic programming concepts that a beginner encounters are a script (a program as a sequence of commands), a loop (with a known number of repetitions), a condition (within a *while* loop), branching (if-then), a subroutine, an event (as a starting point for parallel threads), a variable, and an object (Vaníček, 2019, p. 366). Certain concepts require prior knowledge of other concepts, e.g., the *while* loop is determined by the knowledge of the loop, which is again determined by an understanding of the program as a sequence of commands. Consequently, the acquisition of such concepts depends on an individual's ability to sufficiently comprehend those prior concepts and create a suitable scheme of these concepts and relations between them.
Other studies deal with the issue of mental models in programming. The more complete and veridical one's mental model, the more useful in supporting sophisticated programming it will be (Cañas et al., 1994). As claimed by Ma (2007), students with viable mental models of basic programming concepts perform better in programming tasks than those with nonviable mental models. According to Kesselbacher and Bollin (2019), students who are less successful in solving tasks execute the program more often, as novice programmers have no consistent understanding of basic programming concepts. Meanwhile, students who are more successful in solving tasks execute the program less often. Those students do not rely on program execution to understand their program (Kesselbacher and Bollin, 2019).

2.2. Adaptation of new knowledge

Piaget describes ways of acquiring mental schemes. These are created by an individual from constituent findings during exploration of the surrounding world, and they explain how things work (Piaget, 1930, quoted by Hartl and Hartlová, 2010). If the existing ways of thinking and schemes are adequate in a confrontation with stimuli from an environment, the person will be in a state of balance (equilibrium). However, if the person encounters information which does not fit the existing schemes, a cognitive imbalance will be created. The person will attempt to restore the balance using assimilation, i.e., incorporating new information into the existing schemes. If the new experience does not fit the person´s existing scheme, then the scheme will be altered in a way that incorporates the new information. The process of modification of the scheme is called accommodation (Sternberg, 1999; Nolen-Hoeksema et al., 2009; Piaget and Inhelder, 1997). The concept of accommodation and assimilation is widely used (Hanfstingl et al., 2021; Trigueros, 2019; Dubinsky, 2002) and we will be referring to it in the following section about overcoming misconceptions.

2.3. Misconceptions

Before being taught concepts, pupils have their own conceptions of some phenomena that concepts explain. However, these conceptions are different from the currently accepted scientific concepts (Smith III et al., 1994). In this article, these will be called misconceptions. Misconceptions arise from pupils' prior learning, either in a classroom or from their interaction with the physical and social world (Smith III et al., 1994). Their origin may also be influenced by the way of teaching (Ozgur and Pelitoglu, 2008). From a constructivist point of view, a misconception is not a mistake, but a logical construction based on a consistent but nonstandard theory held by the pupil (Ben-Ari, 2001). Persistent misconceptions can be seen as a novice's effort to extend their existing useful conceptions to instructional contexts, where they turn out to be inadequate (Smith III et al., 1994). As Swidan et al. (2018) claim, holding a misconception is a step towards holding the correct and complete concept.
Constructivist-oriented mathematics teaching methods place a heavy focus on working with mistakes that signify a misconception. In the Czech environment, this is mostly so-called didactic constructivism (Hejný and Kuřina, 2001). Hejný recommends putting the main focus on the reasons for pupils' mistakes (Hejný, 2004). If a mistake is made by a pupil, he/she should be given a different task. The same way of solving that task would lead to an easily recognizable mistake. The pupil should then be allowed to detect the mistake in the original solution. A teacher can thus lead the pupil to a conflict between his/her misconception and the concept (Molnár et al., 2008). The teacher attempts to create a cognitive imbalance for the pupil, resulting in accommodation of new knowledge.

2.4. Misconceptions in programming

According to Ben-Ari (2001), programming is different from other fields as the consequences of misconceptions are exposed immediately, unlike homework feedback which can take up to a week. Sorva (2012) claims that a misconception in programming is usually not universally useless. It may be viable for a particular purpose, but non-viable in general. People may be entirely satisfied with their misconceived notions, even

for a considerable amount of time (Sorva, 2012). However, incorrect and incomplete understanding of programming concepts results in unproductive programming behaviour and dysfunctional programs (Sorva, 2013).

A different view is introduced by Papert (1980). According to him, mistakes benefit pupils because they lead them to study what happened, understand what went wrong, and, through understanding, fix them. He sees mistakes as an intrinsic part of the learning process (Papert, 1980). If a misconception is understood as a pupil´s conception that produces a systematic pattern of mistakes (Smith III et al., 1994), misconceptions and overcoming them can be viewed as a means to obtain a deeper understanding of the topic. Identifying and addressing pupils' misconceptions is a key part of a computer science teacher's competence (Qian and Lehman, 2017).

Many studies have dealt with the issue of misconceptions acquired while learning programming. In his research, Sorva (2012, p. 359) states more than 160 types of misconceptions in programming, dividing them into several groups (e.g., variables, assignment and expression evaluation; subprogram invocations and parameter passing). However, a number of researchers interpret the term misconception in a broader sense to include syntactic mistakes (Brown and Altadmri, 2014) or careless errors (Sekiya and Yamaguchi, 2013).

We accept the view of Qian and Lehman (2017) who perceive misconceptions more narrowly as errors in conceptual understanding. More precisely, misconceptions are actually perceived as sources of mistakes as they cause some wrong or incorrect actions that lead to mistakes. As Qian and Lehman (2017) argue, programming difficulties encountered by students are not always neatly identifiable, and misconceptions may contribute to other kinds of difficulties or mistakes that students have to deal with. Other mistakes which can be observed in programming do not arise from a misconception. Some might be caused by insufficient computational thinking (failure to complete tasks involving algorithmization, decomposition, finding repeating patterns, optimalisation or generalisation), while others might be connected to programming skills.

The occurrence of misconceptions is influenced by the environment in which students are taught programming (Mladenović et al., 2018; Weintrop and Wilensky, 2015). According to Mladenović et al. (2018), students' misconceptions about loops are minimized when using a block-based programming language (Scratch), rather than text-based programming languages (Logo and Python). With more complex tasks, such as using nested loops, the differences become more apparent (Mladenović et al., 2018). On the contrary, Grover and Basu (2017) claim that although it is syntactically easier to put together programs in block-based environments, conceptual difficulties still persist in understanding and using fundamental building blocks of programs such as variables and loops. For learners to understand the concepts which they should use, additional effort and pedagogical strategies are needed (Grover and Basu, 2017).

Certain studies examining misconceptions in programming deal with the frequency of occurrence of observed phenomena (e.g., Sanders and Thomas, 2007). According to Ben-Ari (2001), cataloguing and analysing misconceptions will not be sufficient to improve students' understanding. Instead, research must be undertaken to identify the mental models that cause these specific programming misconceptions, and guidelines must be developed so that teachers can detect and correct the problems (Ben-Ari, 2001).

3. Motivation and research aim

As mentioned above, several studies have already searched for misconceptions in programming and cataloged them. This motivated us to investigate how and to what extent the occurrence of misconceptions in pupils' solutions are influenced by curricula consisting of tasks which lead pupils to develop their concepts. This issue is significant as inappropriately chosen tasks may cause misconceptions which are then difficult to eliminate. Another issue was whether misconceptions still occur in block-based programming when they are minimalized according to Mladenović et al. (2018) and which ones they are.

Hence, we conducted the research to discover which misconceptions occur in beginners' minds while learning one of the first of the more complicated concepts they encounter in programming, i.e., the concept of loop with a known number of repetitions. An additional aim of the research was to discover the factors which influence these misconceptions. Regarding the stated research aim, the following research questions were formulated:

- RQ1: Which misconceptions occur in beginners' minds during their introduction to the concept of loop with a fixed number of repetitions?
- RQ2: How do pupils cope with misconceptions that prevent them from successfully completing a task?
- RQ3: How does the presence of misconception influence how difficultly a pupil finds solution of a particular task?
- RQ4: Which factors concerning the used tasks decrease or increase the existence of misconceptions identified within RQ1?

4. Educational background

The aim was to study the misconceptions that arise when pupils are solving programming tasks which require the use of loops. In this research, these tasks are supposed to be solved by pupils individually and autonomously, without preliminary instruction or accompanying guidance. To support pupils in the learning of concepts, we wanted to create an environment and curricula that would lead pupils through situations in which a misconception of a given concept could arise. We supposed that the generic model of the concept of loop will be formed gradually. We also supposed that pupils would need to overcome (sometimes viable) misconceptions before the correct use of loops is grasped in all its facets.

We chose the concept of loop with a fixed number of repetitions (the *repeat* command) because it is one of easiest concepts for programming beginners to understand. In lower-secondary programming curricula (Kalaš and Miková, 2020; Vaníček et al., 2020), the concept of loop ranks just after the concept of sequence of commands, and before procedures, conditions, events, objects, branching, parameters, and variables.

The command "*repeat* n times" was chosen as a representative of loops. We preferred this *repeat* loop to a similar *for* loop, which uses a variable to count iterations, since beginners might be unable to understand the concept of variable ¬– and this could influence the research results.

4.1. Test environment

We developed a software environment where interactive situational programming tasks can be created. A pupil solves a problem in it by assembling a program from blocks and is given the possibility of testing and debugging his/her program. By collecting data of their program code, researchers are also provided with feedback as to how respondents solved the tasks. We chose a block-based environment to avoid syntax/typo errors that would increase the amount of time needed to correct mistakes caused by misconceptions.
The created environment was implemented as a module in the Bobřík informatiky contest (2022), the Czech edition of Bebras Challenge (Dagiene, 2008) - see Figure 1. There were several reasons for doing this:

- the possibility to use the online environment of this contest and to prepare a set of tasks as a test
- motivation of pupil respondents because they are familiar with this contest
- ethical reasons as Bebras Challenge covers all GDPR issues, e.g., it does not collect any personal data

Our module is based on the Blockly environment as it coincided with the layout of the Bebras test, the use of commands in Blockly having the same philosophy as in Scratch and every launch of pupil's program producing data about it.

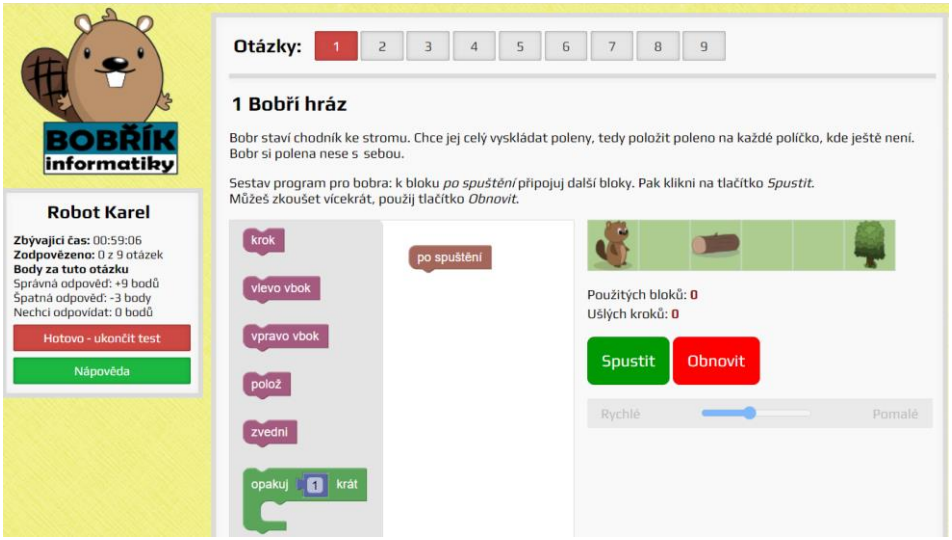

Figure 1. Preview of the test environment and the template simulating the microworld of Karel the Robot used in the Bebras test.

In our implementation, only a limited set of programming commands are available. This prevents pupils from searching the menu and looking for a tool that would simply solve the problem rather than having to think about it ¬– as we can see when pupils solve problems in Scratch (Vaníček, 2019, p. 362).

Pupils create and run their programs any time they want. The sources of information are the instruction blocks, the task texts and the feedback obtained when running their programs. The feedback comprises a visual check of how the programmed sprite is behaving, as well as generated notifications as to whether all requirements have been met. The environment enables us to create sets of consecutive tasks, similar to Hour of Code (2015) activities. The tasks gradually increase in difficulty with the progressive involvement of more complex situations.
In such a setting, pupils can always use the trial and error method to discover the correct use of the *repea*t block (where to place it, how to compose it with other blocks, whether the number of repetitions is correct or must be changed). Pupils can also get a general view by analyzing the situation and using computational thinking. While solving these tasks, pupils create mental models of the given concept by intense activity.
The developed module allowed us to gather programs which pupils created during their attempts to solve the tasks. This allows us to observe how pupils learn to understand concepts they need to solve the problem and where misconceptions appear.
We developed a test with tasks based on the template of the microworld of Karel the Robot (according to Pattis, 1981). In this microworld, a programmed sprite:

- moves in a square grid to the next square
- turns in both directions at a right angle
- detects objects on the square where it is standing
- removes an object from the square
- puts an object on the empty square on which it is standing
- is able to detect an obstacle on the next square in the direction it is facing

A *repeat* structure was added to the commands controlling the sprite, constituting a loop with a fixed number of repeats. No other loop commands were available in the environment.
We chose the "world of Karel the Robot" as it is simple enough to enable pupils to understand and master the basics of language so that they can quickly move on to more complicated tasks. As a result, the time required to gain an understanding of the environment is reduced, enabling pupils to concentrate more intensively on the algorithmic core of set tasks.

4.2. Task design

We created a set of 9 tasks designed for pupils to learn programming basics. These focused on particular programming concepts. The set of tasks involving the use of the loop concept were specifically designed to meet the educational goal of pupils being able to:

- distinguish repeating patterns
- decide whether it is worth using a loop
- distinguish the number of repeats
- find a place where to insert a block (before a loop, inside a loop, after a loop)
- repeat a group of blocks in one loop

Tasks focusing on the programming structure of loop were complemented by introductory tasks of building block sequences (e.g., to order blocks one after another, to edit and reorder blocks). These also fulfilled the requirement of introducing the programming environment and the world of Karel the Robot. Our research was not conducted on those introductory tasks. It would have been difficult to distinguish misconceptions of different kinds (e.g., the relation between the order of blocks in a sequence and the order of execution of these blocks) from other influences such as issues of editing the block program or a new environment.
We avoided incorporating those tasks that require using nested loops as they are significantly more difficult. Moreover, introducing nested loops in the early stages of learning programming would not allow respondents to gain enough experience with the simple loop.
From the concept point of view, individual tasks may be presented in the following way:

1. assembling a program, getting acquainted with the environment
2. assembling a program to move a sprite
3. assembling a longer program
4. using a one-block loop
5. using two blocks in a loop (a task added in the second research phase)
6. using more blocks in a loop with another block before or after the loop
7. using a loop where a robot is given a maximum number of steps
8. using a loop with a hidden repeating pattern
9. using a loop with a more complicated repeating pattern

Starting with Task 4, the length of a program was limited to a certain number of blocks. That led respondents to use a *repeat* block, to gather blocks into the *repeat* block, or to choose a suitable path for the sprite. Certain tasks had more than one correct solution. Task 5 had not been included in the original set of tasks, but it was added in the second phase of the research.
The increasing difficulty of the programming problems can be seen in the set of sample solutions to the first 6 tasks (see Figure 2). These clearly show the aim of gradually increasing the use of the *repeat* block as pupils progressed through the tasks. It is also evident that none of the tasks, including those used to introduce new concepts (task 1 - sequence, task 4 - loop), are trivial. As a consequence, misconceptions could arise. The whole test is available in Czech at https://www.ibobr.cz/test/archiv-pred-spustenim/2021/496 (Bobřík informatiky, 2021).

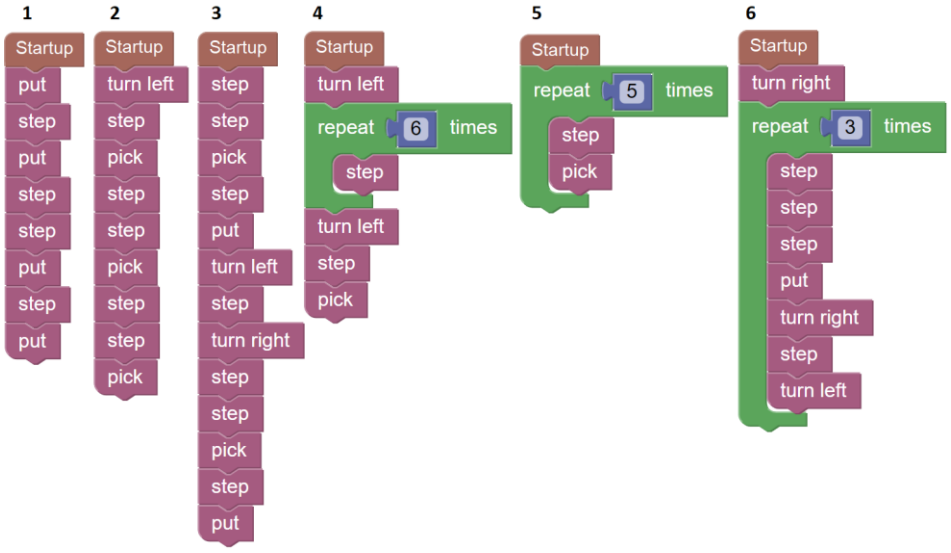

Figure 2. Sample solutions to the first 6 tasks in the set.

5. Methodology

To answer the above-stated research questions, we examined the misconceptions of pupils who were learning the basics of programming and had no prior experience with programming. Pupils were not given any instructions or taught programming concepts, and they did not have the concepts described or explained to them. They were simply assigned a task to solve a problem by assembling a program in the created environment.

Several approaches can be used to discover how pupils learn programming. In many cases, an analysis of completed tasks does not reveal how a pupil progressed. This is therefore supplemented by other techniques, e.g., interviews with pupils, a videorecording of their solution, or eye-tracking (Bednarik and Tukiainen, 2004; Busjahn et al., 2014). However, these methods are time-consuming and only a small number of respondents can be studied as only pure qualitative data are processed. Another method is to automatically process all outputs from all respondents during task-solutions.

We chose a mixed method of research. Moreover, we divided our research into two phases. The original set of tasks was used in the first phase and this set was then adapted for the second phase. In our opinion, the changes made should have helped pupils to overcome the misconceptions we had discovered (one task being added, using explanatory picture and text to accompany a task). However, the core of most of the tasks remained unchanged. Each respondent participated in only one phase of our research.

In the first phase, we answered RQ1 and RQ2. Respondents involved in this phase were also used as the control group for RQ4. Respondents taking part in the second phase were used as the experimental group for RQ4. We used data from both phases to answer RQ3, as the first phase did not provide enough data for statistical evaluation.

We recorded some pupils' screens during the data collection. Those video-recordings

were then coded by open coding, according to Corbin and Strauss (2008), in Atlas.ti software (Friese, 2012; Šimandl and Dobiáš, 2022). In the qualitative analysis, we focused on mistakes or unusual behavior of respondents. We also attempted to identify the misconceptions experienced by individual pupils while working with the loop concept.

Afterwards, we verified the occurrence of the qualitatively found misconceptions using a quantitative method. For this purpose, our test environment records a program assembled by the pupil. This happens every time the pupil presses the Run button to request the program to be run. We gained data that can be analyzed in terms of how many times the pupil asked for the program to be run, the progress a pupil made with his program within a task, and whether that program met all the requirements assigned in the instructions. Those data were then automatically processed in an MS Excel spreadsheet for the purpose of identifying the individual misconceptions of particular pupils.

Where outcomes were unclear, the nature of the obtained data allowed us to implement a detailed process to search for the task solution by a particular respondent. We ultimately used that for all respondents with a quantitatively determined misconception, analyzing their progress in assembling programs and judging individually whether the particular misconception had occurred in those data or not.

5.1. Respondents and the data collection

Our respondents were made up of lower-secondary pupils aged 12-13. At this age, pupils are in the formal operational stage (Piaget, 1970). Unlike older pupils, they are less likely to have encountered programming in lessons at school or during free-time activities.

The test was run online at participating schools under the supervision of the pupils' teachers. In our research, we always worked with whole classes as the data collection was completed during informatics lessons. Respondents were acquired by approaching 15 schools that regularly participate in the Bebras Challenge. Eight schools met our requirement that their under-13 pupils agreed to participate and had not previously encountered programming at school. There were 402 respondents in total: 98 respondents in the first phase, 304 respondents in the second.

The data collection was completed during lessons and pupils were assigned a set of prepared tasks (see 4.2 for details). The time limit for this test was 40 minutes. Respondents were assigned up to 9 tasks (which they were able to solve within the given time limit). We obtained 12 video-recordings of pupils' screens, the average length of each recording being 35 minutes. Using a module which recorded the running programs, we obtained a total of 20,895 assembled programs for analysis. This means every task that was started by a respondent ran 7 times on average.

To adhere to research ethics, we strived to eliminate any negative effects on respondents. Data collection was carried out as a non-competitive online test within the Bebras Challenge environment. Pupils took the test in their own school during lessons, and in most cases the researcher was not present. In compliance with the privacy statement, pupils' personal details were not collected during the test in the Bebras Challenge environment as, in line with GDPR, children under 16 years of age are only allowed to participate anonymously. Video-recordings of pupils' screens captured only

the web browser tab where the test was opened and did not violate pupils' privacy.

5.2. Data analysis

Video-recordings of screens of chosen respondents during the solving of tasks and saved sets of each respondent's created programs allowed us to watch certain kinds of stories in which the respondent was discovering the right solution. It was possible to follow his/her strategies for task solutions from those stories, detect mistakes which he/she made, and analyze whether the mistakes were caused by a misconception or whether there was a different reason for them. Other reasons included a careless mistake while reading task instructions, insufficient familiarity with the environment, or a mistake in space orientation on a graphic screen (typically left-right turning of the sprite).

An example of such a story is one of the respondents' solutions to task 6. In the task, a sprite (a girl) should pick up 3 cans and place them in bins. It cannot crash into a tree by doing so (Figure 3). The right solution is to use one loop with more repeating blocks, with one block before the loop.
One respondent, Vanessa, needed 15 attempts to solve the task. Firstly, she quickly assembled a loop with the right number of repeats that contained more blocks. She then dealt with the number and order of the blocks inside the loop for a longer period of time. She added more blocks inside the loop which forced her to decrease the number of repeats. Finally, she reached the point where the right sequence of blocks was repeated twice inside the loop. In summary, the sprite repeated its activities the wrong number of times (her solution offered an even number of executions but it should have been an odd number according to the instructions). Vanessa realized this after a few attempts. She then removed the excess half of blocks in the loop and changed the number of repeats to the right one.

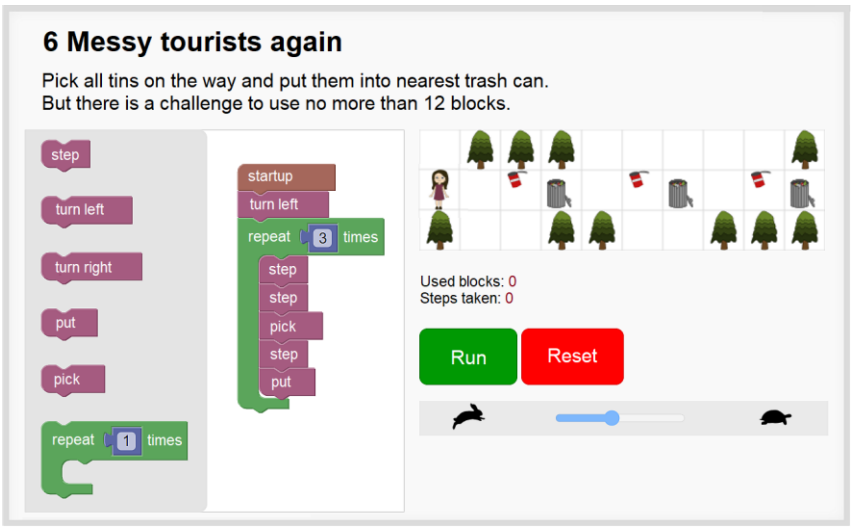

Figure 3. The test environment with instructions for task 6 and the right assembled solution.

As is apparent from the above-mentioned story, we were able to follow not only the strategies for solutions and misconceptions in the sets of programs, but also other mistakes related to the loop and *repeat* block concepts. For example, some respondents:

- incorrectly added before the loop or after the loop the same sets of blocks that had already been placed inside the loop, instead of altering the number of repetitions of the loop (see Figure 4 on the left).
- needed quite a long period of time to order repeating blocks inside the loop
- spent a lot of time removing excess repetitions of the same sequence of blocks inside the loop (see Figure 4 on the right).

These mistakes do not have to be caused by a misconception, but by the fact that the code gets long. Since pupils are still not able to grasp the whole program, they focus only on one part of it. Another reason for a mistake may be insufficient competences in searching for repeating patterns. Pupils are not sure at this moment which parts of the program are repeated, they cannot find such patterns or do not realize the need to search for them.

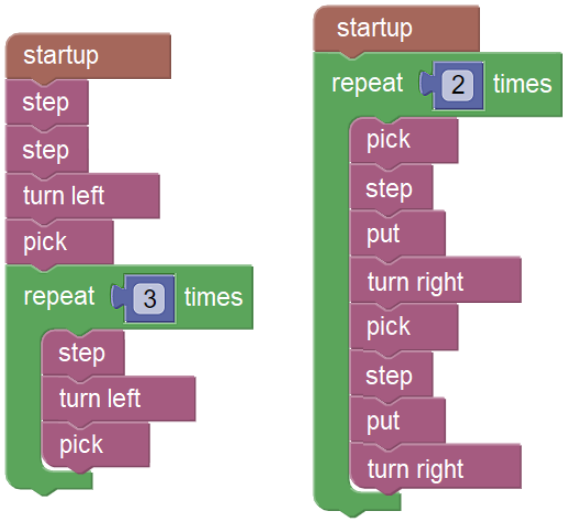

Figure 4. Two programming mistakes not based on a misconception. Additions of the same set of blocks which was inserted inside the loop, before or behind the loop (on the left). An example of repeating the same sequence of blocks inside the loop several times (on the right).

5.2.1. Analysis of data for RQ1

To fully answer RQ1, we examined how to find misconceptions (which we had already encountered in the analysis of the video or chosen assembled programs) in the data of all respondents. For this purpose, we programmed functions in an MS Excel spreadsheet using VBA. Those functions identified misconceptions by individual pupils in the acquired data. The results obtained were recorded in contingency tables. The indicators – which we determined for every misconception – are stated in their description in the

research results. If a pupil first ran the program manifesting the misconception at least three times, we logged this as an occurrence of a pupil's misconception in the particular task.
See the results section for a presentation of the proportion of pupils who manifested a misconception in a particular task.

5.2.2. Analysis of data for RQ2 and RQ3
While searching for answers to RQ2 and RQ3, we used programs assembled during a task solution and data indicating whether the respondent was able to solve the task.

5.2.3. Analysis of data for RQ4
To answer RQ4, we searched for factors that can have an impact on misconceptions. We considered whether the program complexity in the expected task solution, the formulation of task instructions, the order of tasks, or the environment setting could affect the frequency of certain misconceptions. We then changed some task instructions in such a way that the potential factor would eliminate the particular misconception in the next research phase. If it was eliminated, we could claim that the factor under consideration was real. Otherwise, the factor did not influence the misconception. Where there was an increase in the occurrence of the misconception after altering the task, it could be claimed that we had discovered a factor which supports the occurrence of the misconception.
To be able to determine those factors, our research was conducted in two phases, as mentioned above. Research on the original task set was marked as phase one. Phase two was conducted with an altered task set to discover factors that influence misconceptions. For misconception C, phase two was divided into two subphases with task instructions changed in each of them.
Differences in the frequency of misconceptions between individual research phases were tested using a chi-squared test. We tested significance on the level $\alpha = 0.05$ using Statistica and R software.

5.3. Ensuring validity

To eliminate threats to validity of research design, we chose a mixed form of research. In the first research phase, we recorded a total of 15 pupils' screens. These recordings captured pupils' approaches to assigned tasks. A qualitative analysis of the video-recordings led us to identify individual misconceptions.
During the course of the analysis, the authors frequently discussed ways to automatize searching for occurrences of these misconceptions within the data. Several methods were created for that purpose. These methods discovered occurrences of misconceptions with varying reliability. To increase the validity of the research, we individually checked the set of automatically discovered occurrences of pupils' misconceptions. This significantly increased the validity of the research. We are convinced that all of the cases designated as having occurrences of particular misconceptions are valid, and do not involve mistakes caused by other factors.

6. Results

In the following chapters, we demonstrate the relative frequency of the occurrence of the discovered misconceptions in programs assembled by pupils. We concentrated on tasks which were sensitive to a particular misconception, and especially tasks in which the misconception appeared for the first time. We did not consider initial tasks where the concept of a loop does not occur, or tasks focusing mostly on finding an algorithm.

6.1. Discovered misconceptions (RQ1)

We discovered 4 misconceptions related to the loop concept (text in brackets explains what the misconception consists of):

- misconception N - no blocks in a loop (the pupil does not insert any block into the body of the loop)
- misconception O - one block in a loop (the pupil inserts only one block into the body of a loop)
- misconception C - a constant number of repetitions (the pupil does not change this number)
- misconception S - a series of *repeat* blocks (the pupil generates a chain of loops one after another)

These misconceptions will be explained in the following text. Their frequency (see Table 1) is determined in the first task, where they could appear in the first research phase.

Table 1. Proportion of pupils who manifested a particular misconception out of the total number of those who began to solve the task.

Misconception	**Proportion of pupils**	**Total number of pupils**
N (no blocks in a loop)	9.6%	94
O (one block in a loop)	30%	94
C (a constant number of repetitions)	13%	94
S (a series of *repeat* blocks)	25%	93

We expected to find more misconceptions, but were unable to detect them. Either the applied method was insensitive to other misconceptions, or it was unable to distinguish between mistakes caused by other misconceptions and other types of mistakes. An example of one such misconception is pupils not fully understanding how blocks in a loop are executed. Some pupils may not understand that the execution of the last block during the first iteration of the loop is immediately followed by the execution of the first block of the next iteration of the same loop. We were not able to successfully distinguish mistakes caused by this misconception from other mistakes, e.g., from bad pattern recognition.

6.1.1. *Misconception N (no blocks in a loop)*

The first three tasks in our task set can be solved without using a loop. Task 4 "Lost cargo" needs a *repeat* block for it to be successfully completed. The task is shown in Figure 5. Initially, pupils often try to solve the task by using blocks known from previous tasks. Once he/she discovers this is not possible, the only other option is to use a *repeat* block. The pupil does not know the *repeat* block, therefore he/she experiments with it and creates concepts according to the way a particular block works. To successfully solve previous tasks, individual blocks are ordered one after another. Some pupils continue with the same method in the loop, but in such a situation this is a misconception. To be able to overcome it, the pupil has to discover the possibility of putting another block into the *repeat* block. Misconception N lies in the pupil's fallacy that: **"no other block can be put into the *repeat* block"**.

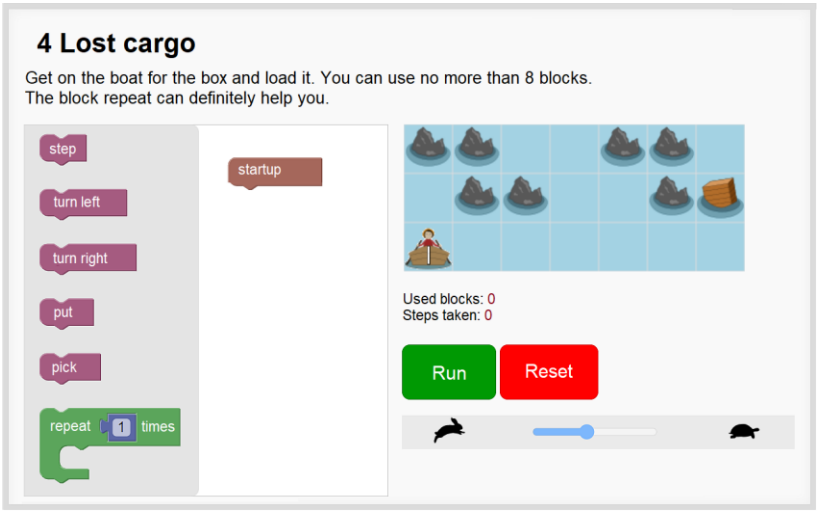

Figure 5. Task 4 in which misconceptions N and C appear.

The misconception indicator here was running the program three times with an empty *repeat* block. Task 4 was the first task to require the *repeat* block. 9.6% of pupils manifested misconception N in that task.

6.1.2. *Misconception O (one block in a loop)*

To solve the first task requiring the use of a loop - task 4 "Lost cargo", exactly one block needs to be put into a *repeat* block. Some pupils manifested the following misconception while programming: **"it is possible to put only one block into a *repeat* block"**. With this misconception, it is possible to solve task 4, but impossible to solve the following tasks, whose successful completion requires more blocks to be put into the *repeat* block.

The first task where misconception O may occur is task 6 (see Figure 3). The misconception indicator here was inserting only one block in the *repeat* block. Misconception O in task 6 occurred in 30% of all pupils who were working on that task. More than a third of these were unable to get rid of misconception O in the following 3

tasks. This proves how difficult it was for a certain number of pupils to overcome misconception O.

6.1.3. Misconception C (a constant number of repetitions)

A *repeat* block, which is chosen by a task solver and then inserted into his/her program, contains a previously adjusted number of repetitions that can be edited. Misconception C lies in the pupil's belief that this **number of repetitions cannot be changed**. Certain pupils tried to solve tasks with the previously adjusted number of repetitions.

The first task that is sensitive to misconception C is task 4 (see Figure 5). The misconception indicator here was the unchanged number of repetitions in a loop as opposed to the implicit number. In task 4, this misconception occurred in 13% of pupils.

6.1.4. Misconception S (a series of repeat blocks)

This misconception lies in the pupil's belief that a **repetition of a block group is achieved by using more *repeat* blocks one after another.** Advanced understanding of the concept of a loop requires the comprehension of how repetitions of a group of commands work. A simple repetition of one block is not enough when solving more complex tasks such as ordering things or walking around a square. Misconception S arises from misunderstanding how the loop works; how and in which order commands in the loop are executed. With this misconception, the pupil adds more *repeat* blocks at the end of the program instead of using only one *repeat* block and putting the group of blocks into the body of the loop (see Figure 6). This may be caused by the fact that pupils were used to adding blocks at the end of the program in the previous tasks. This misconception might be supported by the simplicity of mechanical assembling of *repeat* blocks one after another.

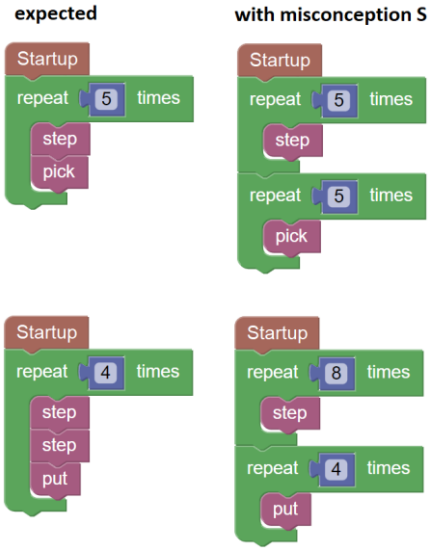

Figure 6. Occurrences of misconception S. The expected solution is on the left and a solution influenced by misconception S on the right. The upper scheme shows a simple situation, the lower scheme shows a more complex situation.

The first task in which misconception S might appear is task 6. The misconception indicator here was the presence of more *repeat* blocks placed one after another with one block in the body of the loop. Misconception S occurred in 25% of respondents.
As is apparent from Table 1, misconceptions O and S appeared much more frequently than the other two misconceptions. We presume that misconceptions O and S are related to a deeper understanding of the loop concept. This coincides with our findings that they occurred in tasks in the second half of the test, where tasks are more complex.

6.2. Variations of coping with misconceptions (RQ2)

If a pupil manifested a misconception which prevents him/her from successfully solving the given task, the pupil may react in two different ways [1] :

- he/she tries to overcome the misconception and go on to successfully solve the task
- he/she gives up his/her attempts to solve the task

The following text examines how pupils cope with the individual misconceptions. The term "to overcome a misconception" means that when solving a task, a pupil first manifested the misconception but then went on to solve the task successfully. As is apparent from Table 2, almost all pupils with misconception C and two thirds of pupils with misconception N were able to overcome those misconceptions. Meanwhile, only one fifth or one seventh of pupils were able to overcome misconception O and misconception S and finish the task.

Table 2. The proportion of pupils who manifested the particular misconception and solved the task from the total number of pupils who manifested that misconception.

Misconception	**Proportion of pupils who overcame the misconception**	**Total number of pupils who manifested that misconception**
N (no blocks in a loop)	67%	9
O (one block in a loop)	14%	28
C (a constant number of repetitions)	92%	12
S (a series of *repeat* blocks)	22%	23

6.3. The impact of the occurrence of a misconception on the difficulty of solving a particular task (RQ3)

1 In the following text, we will omit strategies involving searching for external help (e.g., a teacher, a schoolmate), or using information sources, since pupils involved in the research worked individually and without an opportunity to use information sources.

We were interested in how long it would take a pupil to solve the given task if he/she did or did not develop a misconception. We chose not to measure time because the amount of time needed to solve a task could be influenced by differences in the time required to read instructions or jumping from one task to another. Instead, we counted how many times a pupil asked the computer to check the correctness of the assembled program. We divided all pupils who solved the task correctly into 2 groups: pupils with the particular misconception and pupils without it. We counted the average number of pupils' programs in both groups and compared those numbers (see Table 3).

Table 3. Comparison of average numbers of programs assembled by successful solvers per pupil depending on the presence of the misconception.

Misconception	**Average number of programs assembled by pupils with the misconception**	**Average number of programs assembled by pupils without the misconception**
N (no blocks in a loop)	13.6	5.9
O (one block in a loop)	10.0	3.9
C (a constant number of repetitions)	9.8	5.3
S (a series of *repeat* blocks)	15.3	4.6

We again focused on the first task where the particular misconception appeared. The results in Table 3 suggest the occurrence of the misconception lowered the pupils' ability to solve the given task because it increased the number of checks on assembled programs. This was confirmed by a nonparametric Mann-Whitney U test[2] , according to which, differences are statistically significant at the level of $\alpha=0.05$ for every misconception.

6.4. Factors which influence misconceptions and their impact on the frequency of occurrence of misconceptions (RQ4))

We were seeking factors which influence misconceptions and are related to particular tasks. The occurrence of misconceptions may be influenced by the nature of the task, the formulation of its instructions, and also by tasks which have already been completed by the pupil. We tested whether the frequency of misconceptions can be changed by altering certain task instructions or even by adding another task.
In our interpretation, if the frequency of occurrence decreases, the particular factor reduces the misconception. If the frequency increases, the particular factor strengthens

2 According to a Shapiro-Wilk test, the data were not normally distributed, so a nonparametric Mann-Whitney U test was used.

the misconception. Where no statistical change occurs, the particular task alteration does not behave as a factor. Table 4 shows the probability (p-values) that there is no difference between data from the first and second phase.

Table 4. This shows how implemented changes influenced the occurrence of particular misconceptions in the form of p-values. D indicates a decrease in the frequency of misconceptions. I indicates an increase in the frequency of misconceptions in a particular case. Statistically significant differences are marked with an asterisk ($\alpha = 0.05$).

<table>
<tr><th></th><th>N
(no blocks in a loop)</th><th>O
(one block in a loop)</th><th>C
(a constant number of repetitions)</th><th>S
(a series of repeat blocks)</th></tr>
<tr><td>Additional task</td><td>—</td><td rowspan="2">D 0.00001*</td><td>—</td><td>—</td></tr>
<tr><td>Clarification picture</td><td>D 0.32</td><td rowspan="2">phase 2a:
I 0.034*
phase 2b:
I 0.0044*</td><td>D 0.04*</td></tr>
<tr><td>Change of implicit number</td><td>—</td><td>—</td><td>—</td></tr>
</table>

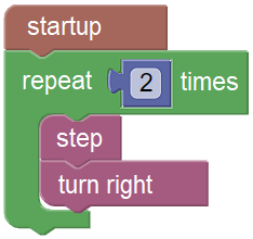

Figure 7. Clarification picture accompanying task 4 instructions in the second research phase.

Contrary to the first research phase, we implemented the following changes into the set of tasks in the second research phase:

1. **Clarification picture:** We added a picture showing an example of a loop with 2 inserted blocks to accompany the instructions for task 4 where the loop appears for the first time (see Figure 7). The following text complemented the picture: “An adjusted *repeat* block may look like this. You can change the number in the blue square.” The picture could not be taken as a hint for solving the particular problem. Through that change we aimed at all four misconceptions.
2. **Additional task:** A large decrease in the number of successful solutions in task 6 compared to task 4 signaled a huge cognitive jump between those tasks. This was caused by the fact that the new concept occurred in a more complex situation (five blocks inside a loop, another block before the loop). Having detected it, we added a new fifth task in the second research phase (see Figure 8). Solving task 5 requires pupils to place two blocks into the loop and no more blocks either before or after the loop. Through that change, we aimed at misconceptions O and S.

3. **Change of implicit number of repetitions in the *repeat* block:** While solving tasks, pupils were assembling a program from blocks which they had selected from the available options. In the first research phase, the *repeat* block had the implicit number of repetitions in all tasks set at 1 (see Figure 5). We changed that number in certain tasks in the second research phase. Through that change we aimed at misconception C.

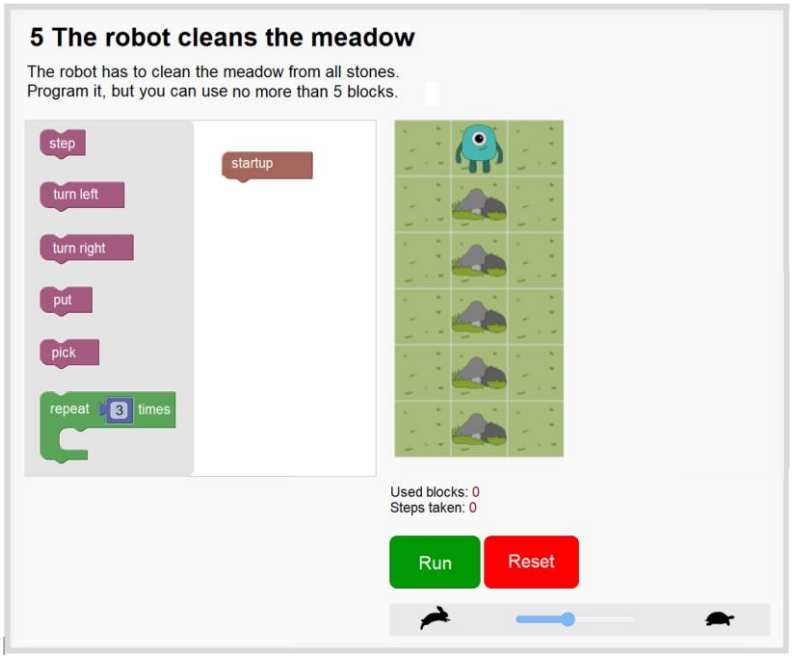

Figure 8. The newly inserted task 5 in the second phase of the research. In the menu on the left, there is a *repeat* block with an implicit number of repetitions different from 1.

6.4.1. Factors which have an impact on misconception N (no blocks in a loop)

A total of 9.6% of pupils manifested misconception N in the first research phase. That number decreased to 6.5% of pupils in the second research phase. The difference between the numbers of pupils who manifested misconception N in the first and second research phase is not statistically significant (p-value = 0.32). Inserting a clarification picture is not a factor which would significantly decrease the occurrence of misconception N.

6.4.2. Factors which have an impact on misconception O (one block in a loop)

We examined whether the implemented alterations to task instructions influence the frequency of occurrence of misconception O. Pupils encountered this misconception for the first time in task 6 in the first research phase. As there was a significant cognitive jump between tasks 4 and 6, it was much more difficult for pupils to identify and overcome the misconception. This led to a longer duration of misconception O. The alterations to task instructions in the second research phase, i.e., adding task 5 and a clarification picture to task 4, resulted in pupils having already encountered misconception O in task 5.

Our alterations led to a decrease in frequency of the occurrence of misconception O – from 30% to 11% of pupils. This difference is statistically significant (p-value = 0.00001) and demonstrates that inserting an appropriate task before the first task in which the misconception occurred and adding a clarification picture to instructions are factors which statistically decrease misconception O. Those measures helped pupils to understand that their concept of *repeat* block with only one inserted block would not lead to the successful solution. Due to this, accommodation was possible, i.e., the pupil's concept changed from "a *repeat* block can have only one inserted block" to "a *repeat* block can have as many inserted blocks as necessary".

In the second research phase, pupils were able to overcome misconception O during the inserted task 5. This led to the achievement in task 6 increasing from 45% of pupils in the first research phase to 74% of pupils in the second research phase.

6.4.3. Factors which have an impact on misconception C (a constant number of repetitions)

For misconception C, we conducted the second research phase in two subphases: 2a and 2b. We decided to influence the frequency of occurrence of misconception C by altering the implicitly given number of repetitions in the *repeat* block. In the first task that was sensitive to misconception C (task 4), we changed the implicit number of repetitions in the *repeat* block from a value of 1 to a value of 3 in the subphase 2a. The frequency of the occurrence of misconception C could also be affected by the clarification picture of the assembled *repeat* block in the instructions for task 4 (see Figure 7). In this picture, we used a different number than 1 for the number of repetitions in the *repeat* block. Task 4 required the sprite to move 6 steps. The settings in this subphase allowed respondents to use two consecutive *repeat* blocks with 3 repetitions, while still remaining within the limit of the maximum number of used blocks. Pupils were able to use the implicit number of repetitions to solve the task correctly, and thus were not forced to overcome misconception C in the task.

Those changes led to an increase in frequency of the occurrence of the misconception from 13% to 24% of pupils in the subphase 2a. This difference is statistically significant (p-value = 0.034). However, we discovered that 18% of pupils solved the task without changing the implicit number of repetitions, i.e., 22 (73%) of the 30 pupils who manifested misconception C managed to find a solution to the task without overcoming this misconception. Those pupils had to overcome the misconception in the tasks that followed.

We decided to conduct another subphase, 2b, again changing the number of repetitions in the *repeat* block from 3 to 4 (which is not a divider of 6) in that task. The settings allowed pupils to add two "one step" blocks behind the loop with number 4, while still

remaining within the limit of the maximum number of used blocks in the task. In addition, we added the following sentence to the instructions for task 4: "You can change the number in the blue block", which could help pupils to overcome that misconception.
In subphase 2b, misconception C occurred in 28% of pupils, indicating another increase in the occurrence of misconception C. These differences between phase 1 and subphase 2b are statistically significant (p-value = 0.0044), while the differences between subphases 2a and 2b are not statistically significant (p-value = 0.45). However, we discovered that 23% of pupils solved the task without changing the implicit number of repetitions, i.e., 38 (81%) of the 47 pupils who manifested misconception C managed to find a solution to the task without overcoming this misconception.
These results are shown in Table 5.

Table 5. Results of misconception C in task 4, where the sprite has to move 6 steps using a loop.

Research phase	**Implicit number of repetitions in the *repeat* block**	**Proportion of pupils - misconception C occurred**	**Proportion of pupils – correct solution without changing the implicit number**	**Total number of pupils who began to solve the task**
1	1	13%	0%	94
2a	3	24%	18%	124
2b	4	28%	23%	167

The results of the second research phase demonstrate that pupils primarily attempt to find the solution without rewriting their concept of work with the *repeat* block. Only a task which cannot be solved without rewriting the wrong concept can lead them to accommodate the new concept of work with the loop, i.e., task 4 in the first research phase and task 5 in the second research phase.
The results also show that setting a different implicit number of repetitions than 1 in the *repeat* block strengthens misconception C. We assume that some pupils accept such a number as the author's intention in the task ¬– and they therefore try to comply with it which strengthens the misconception.

6.4.4. Factors which have an impact on misconception S (a series of repeat blocks)
We were looking for factors that would influence misconception S (the repetition of more blocks is achieved by using more loops one after another) in task 6. The misconception occurred in 24% of pupils in the first research phase and in 17% of pupils in the second research phase. Their comparison showed a statistically significant decrease in the occurrence of the misconception with the p-value = 0.04. It can be claimed that the clarification picture and the text demonstrating the example of a loop with more blocks decreased the occurrence of a chain of loops being wrongly assembled one after another.
We were also investigating the second possible factor – the addition of task 5 which simplifies the transition to the loop with more blocks. Misconception S did not occur in

any respondents in the task 5, but it occurred in subsequent task 6. So it seems that task 5 is not sensitive to misconception S and does not cause it.

7. Discussion

In our research, we discovered other misconceptions than those provided by Sorva (2012, p. 359) in his overview. Sorva examines several misconceptions related to the loop concept. However, he is primarily concerned with misconceptions related to *for* loop variables (e.g., "*for* loop control variables do not have values inside the loop") and evaluating a *while* loop's condition ("*while* loops terminate as soon as the condition changes to false"). This may be due to Sorva (2012) focusing on the CS1 curriculum, whereas we targeted lower-secondary pupils. It may also be due to the fact that the *repeat* loop does not contain a variable, thus it is more sensitive to beginners' misconceptions than the *for* loop.
In this part, we discuss issues connected with overcoming misconceptions:

- Cognitive imbalance leading to accommodation
- Preference for assimilation to accommodation
- Complexity and interconnection of misconceptions
- Opportunities for overcoming misconceptions
- Environments supporting overcoming misconceptions

The support of cognitive imbalance leading to accommodation. While task 6 was successfully solved by less than one half of pupils in the first phase, this proportion increased to three quarters of pupils in the second phase. As there was no alteration to task 6, this percentage change was caused by adding task 5 and using a clarification picture in task 4. Considering the first tasks, which required more than one block to be inserted in a *repeat* block, one third of pupils manifested misconception O in task 6 in the first research phase. This proportion decreased to one tenth in task 5 in the second phase. By inserting task 5, pupils were given a clear indication that this task could not be solved without placing more blocks into the loop. A cognitive imbalance occurred, as described by Sternberg (1999), Nolen-Hoeksema et al. (2009), Piaget and Inhelder (1997). This cognitive imbalance subsequently resulted in pupils accommodating the concept and overcoming misconception O.

Preference for assimilation to accommodation of new knowledge. The cause of misconceptions might lie in the unsuitable generalization of prior experience. In accordance with Smith III et al. (1994), we believe that misconceptions usually originate in prior instruction, as pupils incorrectly generalize prior knowledge to cope with new tasks. Tasks which should lead to accommodation have to be prepared in such a way that they cannot be solved without overcoming the pupil's misconception. If such a possibility exists, pupils will naturally prefer solving the task without being forced to accommodate their existing mental schemes. This is documented in the findings of misconception C in the second research phase.
Considering misconception C, we made it possible to complete task 4 without changing the implicit number of repetitions in the *repeat* block. However, it was actually more difficult to do that than to change the number of repetitions. Nevertheless, most pupils

who manifested this misconception (three quarters in phase 2a, four fifths in phase 2b) opted for the more difficult alternative. These pupils were unable or unwilling to get rid of misconception C and discover the easier alternative based on changing the number of repetitions in the *repeat* block. These results are in line with Sternberg (1999) and Nolen-Hoeksema et al. (2009), who claim that individuals prefer assimilation of the new findings into the existing mental model to accommodation of the old model due to the new findings.

Complexity and interconnection of misconceptions. As opposed to misconceptions N and C, pupils overcame misconceptions O and S with relative difficulty. This may be caused by the fact that misconceptions N and C can be quite easily discovered and removed during program testing. Misconceptions O and S are less specific and more complex as they concern a deeper understanding of how a loop works. Therefore, the pupil might attribute the failure of the program to other reasons. He/she might then attempt to eliminate these other reasons and hence does not focus on the misconception itself. This would explain that misconceptions N and C occurred in the earlier task 4, while misconceptions O and S occurred in the later task 6 – which is more complex.
Another reason may relate to the interconnection of misconceptions O and S. As pupils were inclined to assume that they can put only one block into a *repeat* block (misconception O), they generated a chain of *repeat* blocks (misconception S). In their solutions, each of the *repeat* blocks contained only one block. It can be assumed that many of the pupils did not fully overcome misconception O, leading to the formation of misconception S.

The need for having enough opportunities for overcoming misconceptions. Some misconceptions seem to be very simple and easily corrected. This might give teachers the impression that they do not have to be dealt with in programming education. However, their occurrence in a significant number of pupils and their impact on the difficulty of a task demonstrate the need to bear them in mind. Some teachers might prefer to exclude these misconceptions when formulating tasks and designing a curriculum. Those teachers may assume that when pupils focus on overcoming these misconceptions, they lose the time and energy needed to solve real problems that would develop their ability to program. However, without encountering misconceptions, pupils cannot improve their mental models of concepts. Moreover, we can easily fail to solve real-world problems without correctly built generic mental models of concepts. This is in line with Sorva (2012), who claims that an incomplete understanding of programming concepts results in unproductive programming behavior.

The need for suitable environments supporting overcoming misconceptions. In Table 2, there are high proportions (two thirds and nine tenths) of pupils who overcome misconceptions N and C. This shows that the majority of pupils who manifested these misconceptions overcame them in the very same task, leading us to believe that the wording of the questions and the nature of the environment were appropriately selected. Consequently, pupils quickly overcame a particular misconception and gained an understanding of the loop concept more quickly. If the task is more complex and contains more problematic areas for the pupil, it will significantly decrease the probability of him/her overcoming the misconception.

Environments like “Hour of Code” might be considered helpful. This allows pupils to assemble a program from blocks in a graphic environment. Provided tasks are selected appropriately, this environment provides a number of situations where pupils can develop misconceptions and quickly overcome them. In accordance with the theory of generic mental models (Hejný, 2012), this enables pupils to create a generic model of a particular concept more easily.

8. Conclusion

In this study, we researched aspects concerning the creation of concepts for learning programming at an age of 12-13. We looked into comprehension of the loop concept in a set of programming tasks in a block-based environment. We researched pupils who had not previously learned programming at school. These pupils solved tasks that were assigned in the form of an online test. Pupils had no preparatory or accompanying instruction or explanation. We looked for misconceptions that prevent beginners from successfully solving such tasks.
We were able to detect four misconceptions:

- misconception N – no command is inserted into a *repeat* block
- misconception O – only one command is possible to insert into a *repeat* block
- misconception C – the number of repetitions in a loop cannot be changed
- misconception S – the repetition of a number of commands is achieved by using a number of loops one after another

Our results show that misconceptions N and C are easily overcome, while misconceptions O and S are not.
The presence of a misconception makes it more difficult for a pupil to solve a task. It also significantly increases the pupil’s need to ask the computer to check the correctness of the assembled program. We discovered factors which influence some of these misconceptions:

1. inserting a picture clarifying programming code of two blocks in a loop reduces misconceptions O and S
2. setting a different implicit number of repetitions than 1 in a *repeat* block strengthens misconception C

It appears that an analysis of pupils’ work can reveal their misconceptions and that programming lessons could be improved by adding:

- tasks in which a teacher would easily detect a pupil’s misconception
- tasks which would be more resilient to the occurrence of misconceptions
- tasks which would enable pupils with misconceptions to accommodate their mental model and thus overcome the misconception
- tasks focusing on one specific misconception rather than on a combination of them
- tasks which are impossible to solve without overcoming the misconception which the task is focused on

One limitation of this study might be that we were not always able to clearly distinguish between mistakes caused by a misconception and other kinds of mistakes. The examined occurrences may not be cases of misconceptions on all occasions. There may be mistakes caused by a pupil not knowing how to write his/her idea in the particular language. Despite the detailed analysis of a particular program, we were not always sure of the origin of the mistake on every occasion.

A further limitation could be the specific context the research was carried out in. Although a teacher was present in the classroom, he/she did not interfere in the pupils' learning process (he/she did not offer any hints on how to solve the tasks or discuss pupils' mistakes with them). Pupils received all information and feedback from the test environment.

Another limitation could be that data collection was completed online. The researchers were not present in the classroom. If a teacher was not thorough and did not follow our instructions, this may have led to negative occurrences such as cheating, distorting the results of our research. The absence of a researcher was deliberate as we did not want to make pupils nervous due to the presence of a stranger in the classroom. We also wanted to respect the ethical principles of research by protecting the anonymity of research participants. Since pupils' achievement in tasks was not to be assessed, we believe that there were no negative occurrences such as cheating.

A final limitation of the study is the risk that some pupils teach themselves programming out of school. However, we assume that not many pupils in that particular age group actually do so. This would only have had an impact on the results of RQ2.

In future research, it would be worth focusing on more difficult programming concepts such as conditions, variables, or parameters in functions. Research could concentrate on methods which would manage the occurrence of misconceptions by using a suitable choice of instructions in programming tasks. This could involve deliberately initiating the occurrence of misconceptions that the pupil would have to overcome.

Acknowledgments

This research was covered from projects TAČR TL03000222 – The development of Informatics thinking by means of situational algorithmic problems.

References

Ackermann E. (2010). Constructivism(s): Shared roots, crossed paths, multiple legacies. In: Clayson, J. E., Kalaš I. (Eds.), *Constructionism 2010: Constructionist approaches to creative learning, thinking and education: lessons for the 21st century: proceedings for Constructionism 2010*. Comenius University, Bratislava. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.473.6201&rep=rep1&type=pdf

Bednarik, R., Tukiainen, M. (2004). Visual attention tracking during program debugging. In: *Proceedings of the third Nordic conference on Human-computer interaction* (*NordiCHI '04*). Association for Computing Machinery, New York, NY, USA, 331–334. doi: 10.1145/1028014.1028066

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of*

Computers in Mathematics and Science Teaching. 20(1), 45-73. https://www.learntechlib.org/primary/p/8505/

Bers, M.U. (2017*). Coding as a playground: programming and computational thinking in the early childhood classroom*. Routledge, New York.

Bers, M.U., González-González, C., Armas–Torres, B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education*. 138, 130-145. doi: 10.1016/j.compedu.2019.04.013

Bobřík informatiky (2021). Bloky: a set of educational programming tasks. In*: Beaver of Informatics.* Jihočeská univerzita v Českých Budějovicích, České Budějovice, Czechia. https://www.ibobr.cz/test/archiv-pred-spustenim/2021/496

Bobřík informatiky (Beaver of Informatics) (2022). Jihočeská univerzita v Českých Budějovicích, České Budějovice, Czechia. https://www.ibobr.cz/english-uk

Brown, N.C.C., Altadmri, A. (2014). Investigating novice programming mistakes: educator beliefs vs. student data. In: *Proceedings of the tenth annual conference on International computing education research* (*ICER '14*). Association for Computing Machinery, New York, NY, USA, 43–50. doi: 10.1145/2632320.2632343

Busjahn, T., Schulte, C., Sharif, B., Simon, Begel, A., Hansen, M., Bednarik, R., Orlov, P., Ihantola, P., Shchekotova, G., Antropova, M. (2014). Eye tracking in computing education. In: *Proceedings of the tenth annual conference on International computing education research* (*ICER '14*). Association for Computing Machinery, New York, NY, USA, 3–10. doi: 10.1145/2632320.2632344

Cañas, J.J., Bajo, M.T., Gonzalvo, P. (1994). Mental models and computer programming. *International Journal of Human-Computer Studies*. 40(5), 795-811. doi: 10.1006/ijhc.1994.1038

Chao, P.-Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*. 95, 202-215. doi: 10.1016/j.compedu.2016.01.010

Corbin, J.M., Strauss, A.L. (2008). *Basics of qualitative research: techniques and procedures for developing grounded theory.* 3rd ed., SAGE Publications, Los Angeles.

Dagiene, V. (2008). The BEBRAS Contest on Informatics and Computer Literacy – Students' Drive to Science Education. In: J*oint Open and Working IFIP Conference. ICT and Learning for The Next Generation*. 214–223. https://www.bebras.org/sites/default/files/documents/publications/DagieneV-2008.pdf

Dagienė, V., Futschek, G. (2019). On the way to constructionist learning of computational thinking in regular school setting. *Constructivist foundation*. 14(3), 231–233. https://constructivist.info/14/3/231

Dubinsky, E. (2002). Reflective Abstraction in Advanced Mathematical Thinking. In: Tall, D. (Ed.), *Advanced Mathematical Thinking. Mathematics Education Library*. Springer, Dordrecht, 95-126. doi: 10.1007/0-306-47203-1_7

Friese, S. (2012). *Qualitative data analysis with ATLAS.ti*. SAGE Publications, London.

Gander, W. (2014). Informatics and General Education. In: Gülbahar, Y., Karataş, E. (Eds.), I*nformatics in Schools. Teaching and Learning Perspectives. ISSEP 2014. Lecture Notes in Computer Science, vol 8730*. Springer, Cham, 1-7. doi:

10.1007/978-3-319-09958-3_1
Grover, S., Basu, S. (2017). 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* (*SIGCSE '17*). Association for Computing Machinery, New York, NY, USA, 267–272. doi: 10.1145/3017680.3017723
Hanfstingl, B., Arzenšek, A., Apschner, J., Gölly, K.I. (2021). Assimilation and Accommodation: A Systematic Review of the Last Two Decades. *European Psychologist*. 27(4), 320-337. doi: 10.1027/1016-9040/a000463
Hartl, P., Hartlová, H. (2010). *Velký psychologický slovník (Great psychological dictionary)*. Portál, Praha.
Hejný, M. (1990). *Teória vyučovania matematiky 2 (Theory of mathematics education 2)*. Slovenské pedagogické nakladateľstvo, Bratislava.
Hejný, M. (2004). Mechanizmus poznávacího procesu (Mechanism of cognitive process). In: Hejný, M., Novotná, J., Stehlíková, N. (Eds.), *Dvacet pět kapitol z didaktiky matematiky (25 chapters of didactics of mathematics)*. Univerzita Karlova, Praha, 23–42. https://mdisk.pedf.cuni.cz/SUMA/MaterialyKeStazeni/PublikaceKnihy/25KapitolZDM.pdf
Hejný, M. (2012). Exploring the cognitive dimension of teaching mathematics through a scheme-oriented approach to education. *Orbis scholae*. 6(2), 41–55. https://karolinum.cz/data/clanek/5036/OS_2_2012_final.41-55.pdf
Hejný, M., Kuřina, F. (2001). *Dítě, škola a matematika: konstruktivistické přístupy k vyučování (Child, school and mathematics: constructivist approaches to teaching)*. Portál, Praha.
Hour of code (2015). Educational web. https://hourofcode.com/
Kalaš, I., Miková, K. (2020). *Základy programování ve Scratch pro 5. ročník ZŠ. Textbook*. Jihočeská univerzita v Českých Budějovicích, České Budějovice. https://imysleni.cz/ucebnice/zaklady-programovani-ve-scratchi-pro-5-rocnik-zakladni-skoly
Kesselbacher, M., Bollin, A. (2019). Discriminating Programming Strategies in Scratch: Making the Difference between Novice and Experienced Programmers. In: *Proceedings of the 14th Workshop in Primary and Secondary Computing Education* (*WiPSCE'19*). Association for Computing Machinery, New York, NY, USA. doi: 10.1145/3361721.3361727
Liao, Y.-k. (2000). A Meta-analysis of Computer Programming on Cognitive Outcomes: An Updated Synthesis. In: Bourdeau, J., Heller, R. (Eds.), *ED-MEDIA 2000 World Conference on Educational Multimedia, Hypermedia & Telecommunications. Advancement of Computing in Education (AACE)*, Montreal, Canada, 598-604. https://www.learntechlib.org/primary/p/16132/
Lye, S.Y., Koh, J.H.L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior*. 41, 51-61. doi: 10.1016/j.chb.2014.09.012
Ma, L. (2007). *Investigating and Improving Novice Programmers' Mental Models of Programming Concepts*. PhD Dissertation. University of Strathclyde, Glasgow.
Mladenović, M., Boljat, I., Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education*

and Information Technologies. 23, 1483–1500. doi: 10.1007/s10639-017-9673-3
Molnár, J., Schubertová, S., Vaněk, V. (2008). *Konstruktivismus ve vyučování matematice (Constructivism in the teaching of mathematics)*. Univerzita Palackého v Olomouci, Olomouc. http://esfmoduly.upol.cz/publikace/molnar.pdf
Nolen-Hoeksema, S., Frederickson, L.B., Loftus, G.R., Wagenaar, W.A. (2009). *Atkinson & Hilgard's Introduction to Psychology*. Cengage Learning, Andover.
Ozgur, S., Pelitoglu, F.C. (2008). The Investigation of 6th Grade Student Misconceptions Originated from Didactic about the "Digestive System" Subject. *Educational Sciences: Theory & Practice*, 8(1), 149-159.
Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books, New York, NY, USA.
Pattis, R. E. (1981). *Karel the Robot: Gentle Introduction to the Art of Programming with Pascal.* John Wiley & Sons.
Piaget, J. (1970). *Science of Education and the Psychology of the Child*. The Viking Press: New York.
Piaget, J., Inhelder, B. (1997). *Psychologie dítěte (Psychology of the child)*. Translated by Vyskočilová. Portál, Praha.
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. doi: 10.1145/3077618
Román-González, M., Pérez-González, J.-C., Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*. 72, 678-691. doi: 10.1016/j.chb.2016.08.047
Sanders, K., Thomas, L. (2007). Checklists for grading object-oriented CS1 programs: concepts and misconceptions. In: *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education* (*ITiCSE '07*). Association for Computing Machinery, New York, NY, USA, 166–170. doi: 10.1145/1268784.1268834
Sekiya, T., Yamaguchi, K. (2013). Tracing quiz set to identify novices' programming misconceptions. In: *Proceedings of the 13th Koli Calling International Conference on Computing Education Research* (*Koli Calling '13*). Association for Computing Machinery, New York, NY, USA, 87–95. doi: 10.1145/2526968.2526978
Scherer, R., Siddiq, F., Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology*. 111(5), 764-792. doi: 10.1037/edu0000314
Smith III J.P., diSessa A.A., Roschelle, J. (1994). Misconceptions Reconceived: A Constructivist Analysis of Knowledge in Transition. *Journal of the Learning Sciences*. 3(2), 115-163. doi: 10.1207/s15327809jls0302_1
Sorva, J. (2012). *Visual Program Simulation in Introductory Programming Education*. PhD Dissertation. Aalto University, Espoo, Finland.
Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education.* 13(2). doi: 10.1145/2483710.2483713
Sternberg, R.J. (1999). *Cognitive psychology*. Harcourt Brace, Orlando.
Swidan, A., Hermans, F., Smit, M. (2018). Programming Misconceptions for School Students. In: *Proceedings of the 2018 ACM Conference on International*

Computing Education Research (*ICER '18*). Association for Computing Machinery, New York, NY, USA, 151–159. doi: 10.1145/3230977.3230995

Šimandl, V., Dobiáš, V. (2022). Analýza dat při tvorbě zakotvené teorie pomocí software atlas.ti (Using atlas.ti software for data analysis through the construction of grounded theory). *Paidagogos*, 2021(1), 131-156. http://www.paidagogos.net/issues/2021/1/article.php?id=8

Trigueros, M. (2019). The development of a linear algebra schema: learning as result of the use of a cognitive theory and models. *ZDM Mathematics Education*, 51, 1055–1068. doi: 10.1007/s11858-019-01064-6

Vaníček, J. (2019). Early Programming Education Based on Concept Building. *Constructivist foundation*. 14(3), 360-372. https://constructivist.info/14/3/360.vanicek

Vaníček, J., Nagyová, I., Tomcsányiová, M. (2020*). Programování ve Scratch pro 2. stupeň základní školy. Textbook.* Jihočeská univerzita v Českých Budějovicích, České Budějovice. https://imysleni.cz/ucebnice/programovani-ve-scratchi-pro-2-stupen-zakladni-skoly

Weintrop, D., Wilensky, U. (2015). Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs. In: *Proceedings of the eleventh annual International Conference on International Computing Education Research* (*ICER '15*). Association for Computing Machinery, New York, NY, USA, 101–110. doi: 10.1145/2787622.2787721

Xia, B. S. (2017). A Pedagogical Review of Programming Education Research: What Have We Learned. *International Journal of Online Pedagogy and Course Design.* 7(1), 33-42. doi: 10.4018/IJOPCD.2017010103

About authors

Jiří Vaníček is an associate professor and the head of the Department of Informatics in the Faculty of Education at the University of South Bohemia in České Budějovice in the Czech Republic. His area of interest is Informatics education in primary and lower secondary schools and early age programming. He is an author of 7 textbooks about information technology and programming. Between 2017 and 2020 he was a head of the strategic project PRIM developing new Czech national informatics curricula. He has been organizing Bebras Challenge for 14 years and he is a representative of the Czech Republic in the International Bebras Committee.

Václav Dobiáš is an assistant professor in the Department of Informatics in the Faculty of Education at the University of South Bohemia in České Budějovice in the Czech Republic. His research activities focus on the digital divide and computational thinking.

Václav Šimandl is an assistant professor in the Department of Informatics in the Faculty of Education at the University of South Bohemia in České Budějovice in the Czech Republic. His research activities focus on the area of Informatics education in lower secondary schools and programming. He has been organizing the Czech Bebras Challenge for 8 years.