UNIVERZITA MATEJA BELA
Fakulta prírodných vied, Katedra informatiky

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta přírodovědně-humanitní a pedagogická

NEW PERSPECTIVES IN INFORMATICS EDUCATION
International Proceedings on Teaching Informatics

DidInfo 2025

30th Conference Year

ISBN 978-80-557-2249-8
EAN 978055722498
DOI 10.24040/2025.9788055722498

BANSKÁ BYSTRICA, SLOVAKIA
2025

Programme Committee Chairs:
doc. Ing. Jarmila Škrinárová, PhD. *Univerzita Mateja Bela, SK*
doc. PaedDr. Jiří Vaníček, PhD. *Jihočeská Univerzita, CZ*

Programme Committee:
doc. RNDr. Gabriela Andrejková, CSc. *Univerzita Pavla Jozefa Šafárika, SK*
Mgr. Jan Berki, PhD. *Technická univerzita v Liberci, CZ*
doc. RNDr. Miroslava Černochová, CSc. *Univerzita Karlova, CZ*
prof. Dr. Valentina Dagiene *Vilnius University, LT*
Mgr. Adam Dudáš, PhD. *Univerzita Mateja Bela, SK*
PaedDr. Ján Guniš, PhD. *Univerzita Pavla Jozefa Šafárika v Košiciach, SK*
doc. Mgr. et Mgr. Marie Hubálovská, PhD. *Univerzita Hradec Králové, CZ*
prof. RNDr. Štěpán Hubálovský, PhD. *Univerzita Hradec Králové, CZ*
prof. Dr. Mirjana Ivanovic *University of Novi Sad, SR*
Ing. Jana Jacková, PhD. *Katolícka univerzita v Ružomberku, SK*
prof. RNDr. Ivan Kalaš, PhD. *Univerzita Komenského, SK*
doc. RNDr. Zuzana Kubincová, PhD. *Univerzita Komenského, SK*
doc. RNDr. Gabriela Lovászová, PhD. *Univerzita Konštantína Filozofa, SK*
Ing. Božena Mannová, PhD. *České vysoké učení technické, CZ*
RNDr. Alžbeta Michalíková, PhD. *Univerzita Mateja Bela, SK*
RNDr. Pavel Pešat, PhD. *Univerzita J. E. Purkyně, CZ*
Prof. Dr. Kate Sanders *Rhode Island College, USA*
prof. Ing. Veronika Stoffová, CSc. *Trnavská univerzita v Trnave, SK*
doc. RNDr. Petr Šaloun, PhD. *Technická univerzita Ostrava, CZ*
doc. RNDr. Ľubomír Šnajder, PhD. *Univerzita Pavla Jozefa Šafárika, SK*
doc. RNDr. Pavel Töpfer, CSc. *Univerzita Karlova, CZ*
doc. Ing. Ľudovít Trajteľ, PhD. *Univerzita Mateja Bela, SK*
Dr. Livia Tudor *Petroleum-Gas University, RO*
doc. PaedDr. Patrik Voštinár, PhD. *Univerzita Mateja Bela, SK*

Organizing Committee:
doc. PaedDr. Patrik Voštinár, PhD.
Mgr. Adam Dudáš, PhD.
RNDr. Alžbeta Michalíková, PhD.
Katarína Gavaldová

Reviewers:
doc. RNDr. Gabriela Andrejková, CSc. *Univerzita Pavla Jozefa Šafárika, Košice, SK*
Mgr. Jan Berki, PhD. *Technická Univerzita v Liberci, CZ*
Ing. Jindra Drábková, PhD. *Technická Univerzita v Liberci, CZ*
Mgr. Adam Dudáš, PhD. *Univerzita Mateja Bela, Banská Bystrica, SK*
PaedDr. Ján Guniš, PhD. *Univerzita Pavla Jozefa Šafárika v Košiciach, SK*
Ing. Dana Horváthová, PhD. *Univerzita Mateja Bela, Banská Bystrica, SK*
PaedDr. Roman Hrušecký, PhD. *Univerzita Komenského, Bratislava, SK*
doc. RNDr. Miroslav Iliaš, PhD. *Univerzita Mateja Bela, Banská Bystrica, SK*
Ing. Jana Jacková, PhD. *Katolícka univerzita v Ružomberku, SK*
doc. RNDr. Ľudmila Jašková, PhD., *Univerzita Komenského, Bratislava, SK*
prof. RNDr. Ivan Kalaš, PhD. *Univerzita Komenského, SK*
doc. Mgr. Ján Karabáš, PhD. *Univerzita Mateja Bela, Banská Bystrica, SK*
doc. RNDr. Zuzana Kubincová, PhD., *Univerzita Komenského, Bratislava, SK*

Editors: Mgr. Adam Dudáš, PhD., RNDr. Alžbeta Michalíková, PhD., doc. PaedDr. Patrik Voštinár, PhD., doc. Ing. Jarmila Škrinárová, PhD.

ISBN: 978-80-557-2249-8
EAN: 978055722498
DOI: 10.24040/2025.9788055722498
Electronic, conference proceedings

These conference proceedings were supported by KEGA project number 010TTU-4/2025 with the title Development of pupils' computational and algorithmic thinking through the effective integration of modern technologies into the teaching of computer science and mathematics.

Foreword

We are meeting at the Faculty of Natural Sciences of Matej Bel University in Banská Bystrica for the round 30th DidInfo 2025 conference. The conference with a beautiful tradition is focused on new methods and practices of teaching computer science. It is an honour to welcome all participants.

The main objective of the conference is to act on the comprehensive development of the personalities of the student and the teacher in the subject of computer science and multidisciplinary through the subject of computer science. Teaching computer science has been the domain of secondary schools in the past. Slovakia is one of the first countries to introduce the teaching of computer science in primary schools and this step was justified. We have moved on, and we know that for today's technological world it is necessary to lead children towards STEM (Science - Technology - Engineering - Mathematics) education. As an example, with the advent of artificial intelligence tools for children (especially machine learning), we are teaching them how to create learned modules and use them in programming languages. We are trying to use the big language models utilized by GenAI in education. We are teaching how to use GenAI in schools and embed them into school education and how children should work with them.

Key reasons why it is extremely important to teach children computer science in schools and discuss it at the conference:

- Computer science develops logical thinking and teaches children to think analytically, logically and computationally. Writing code or solving algorithmic problems helps them develop the ability to solve problems step by step.
- In computer science, children can create their own programs, apps or games. This encourages their creativity, imaginative and innovative thinking.
- It is important that children understand technology not only as users but also as creators.
- Computer science develops critical thinking skills. It is important for them to be able to distinguish reliable sources of information from misinformation.
- Teaching computer science provides all children with equal opportunities to master skills that will be critical to their careers and lives.
- Computer science helps children to be smart and to better understand the world around them.
- Computer science is motivating for children, they usually like it, and the understanding and satisfaction in their eyes gives us a reason to be better teachers.

We place more emphasis on basic human values (good relationships, friendships, teamwork, selflessness, ethical behavior, etc.). We do this at the same time as solving computer science problems, e.g. in the First Lego League competition, competitors get more points for demonstrating these values. The so-called “wellbeing” in schools becomes more and more relevant. It is a state in which physical, cognitive, emotional, social and spiritual potential can be

fully developed in a supportive and stimulating environment, and a full and happy life can be lived with others.

I am delighted to welcome among us foreign personalities who have accepted the invitation and prepared engaging lectures: prof. Daniela Tuparova from the South-West University Neofit Rilski from Bulgaria, who deals with didactics of programming, challenges and solutions, and prof. Irena Nancovska Šerbec from the University of Ljubljana, Slovenia, who also specializes in didactics of computer science and will introduce us to current practices with an emphasis on interdisciplinary approach in the European context.

We wish the conference the fulfilment of its goals, the success of the active participants and lots of inspiration for the next year!

This international conference proceedings contains a selected 24.8% of the contributions that were recognized by the programme committee as the best of this year's conference.

March 2025

doc. Ing. Jarmila Škrinárová, PhD.

Programme committee chair

DidInfo 2025 conference

Table of contents

INVITED CONTRIBUTIONS

Teaching Informatics in Slovenia: Current Practices in a European Context and a Cross-Disciplinary Approach

Irena Nančovska-Šerbec
Faculty of Education
University of Ljubljana
Slovenia
Irena.Nancovska@pef.uni-lj.si

ABSTRAKT

Informatics education across Europe varies in structure, integration, and accessibility. While some countries embed it in broader STEM curricula, others, like Slovenia, Lithuania, and Bulgaria, treat it as a separate subject. In Slovenia, informatics is elective in primary school and compulsory only in the first year of upper secondary education. In Croatia, it is compulsory from grade 5 onward, while Spain, Italy, and Cyprus integrate it through technology or digital literacy courses. These differences contribute to variability in informatics competencies, with limited early exposure, inconsistent curriculum coverage, and insufficient teacher training as common challenges.

This study, part of the Digital First project, examines Slovenia's informatics education in a European context. Digital First seeks to enhance curricular coherence, strengthen computational thinking, and support interdisciplinary integration, ensuring progressive skill development and improved teacher training.

As an example, Slovenian educators introduce computational thinking through unplugged methods, leveraging culturally responsive learning in language and the arts to make abstract informatics concepts more tangible. This approach, explored in the INCTCORPS project, aligns with Bruner's enactive, iconic, and symbolic learning stages, helping students progress from physical engagement to abstract reasoning while strengthening motivation and conceptual understanding.

Findings suggest that interdisciplinary strategies enhance engagement and accessibility, particularly in low-tech environments, reinforcing the need for curriculum adjustments and targeted teacher training to support sustainable informatics education.

Education in Programming at Bulgarian Lower Secondary Schools – Challenges and Solutions

Daniela Tuparova
South-West University "Neofit Rilski",
Bulgaria
ddureva@swu.bg

Brief history of teaching programming in bulgarian schools

Education in programming in Bulgarian schools has a long history and is accompanied by dynamic changes, inclusion and exclusion from school curricula and syllabi. [1] It began in the late 1960s with the introduction of programming classes in specialized mathematics high schools. In the period 1979-1999, the Problem group in education at Ministry of Education introduced experimental integrated teaching programming using Logo in the lower secondary school level [2]. In the 1986/1987 school year, a mandatory subject of Informatics was introduced in high school (grades 10 and 11), which included the study of algorithms and programming in Basic. In the period 1991-2000, changes were observed in the curricula and syllabi, in which the hours related to programming education were reduced. In 2000/2001, two subjects were established in high school grades 9 and 10 - Informatics and Information Technologies, with programming being part of the subject of Informatics. From 2006/2007, the subject of Information Technologies was introduced in the lower secondary school level, but without topics related to programming. In 2018/2019, the subject of Computer Modelling was introduced in primary school (grades 3 and 4), focusing mainly on programming in a block environment. As a natural continuation of programming education at the primary level in 2021/2022, a new subject Computer Modelling and Information Technology (CMIT) has been introduced at the lower secondary level (grades 5-7), with 18 hours added and a programming module introduced.

Teaching programming in primary school level

The introduction of programming at the primary and lower secondary levels poses various challenges. The presentation of education in programming at the lower secondary level requires description of the curricula and challenges for the primary school level.

Education in programming at the primary school level is implemented in grades 3 and 4 (9–10-year-old students) with one hour per week. A block programming environment is used, the most commonly used environment being Scratch. In some schools, optional courses related to programming and robotics are also offered. A detailed presentation of the programming education ecosystem in Bulgarian primary schools is presented in [1] and [3].

The curricula (https://www.mon.bg/nfs/2024/01/up_iii_km_260124.pdf , https://www.mon.bg/nfs/2024/01/up_iv_km_260124.pdf) include the main topics as follows:

- Content of computer modelling subject in 3rd grade - Information, digital devices, digital devices control and text and number entering; Safety in a digital environment; Constructing sequential and looping activities; Visual programming environment; Working with text and sound in visual programming environment; Creation of animated projects.
- Content of computer modelling subject in 4th grade - Information, digital devices, digital devices control and text and number entering; Digital identity and safety in digital environments; Working in virtual programming environment; Management of programmable devices; Development of educational games; Development of digital project.

Main challenges in the curricula are high abstraction of the learned concepts as file, information, algorithm, digital identity, loop and branching algorithms, logical and arithmetic operators, random

number, variables etc. Also, the curricula require to be used of mathematical concepts that are part of the math curricula in next years: coordinates and coordinate system; negative numbers; measurement of angles, random numbers. These challenges can be overcome using didactic methods such as:

- Explaining concepts by examples.
- Focus on the properties of the concepts.
- Use of analogies and examples from everyday life and tales.
- Use of fun elements - challenges, riddles, anecdotes, puzzles.
- Use of computer educational games, simulations or demonstrations.
- Interactive tests and electronic textbooks.
- Use of unplugged activities.
- Problem solving
- Experiments with code prepared in advance and completing of "half baked" code.

Programming as a part of new subject Computer modelling and information technology in 5th, 6th and 7th grades

The curriculum in the lower secondary school level is built on a spiral model, including several basic modules, which are expanded every year. The training is carried out within 51 hours per year.

The basic modules in the fifth grade are eight and are arranged in the following sequence: Computer system and information technology; Internet; Audio and video; Computer graphics; Computer modelling; Word processing; Spreadsheets; Presentations

In the fifth grade, education in programming continues with work in a block programming environment. The content of module "Computer modelling" is focussed on creating graphics with a learned block programming environment (design of own computer character; using code for drawing studied plane figures in mathematics); creating and using own blocks or subroutines; development of educational project with the tools of the studied language for block programming (application of algorithms for the implementation of the following activities: exchange of values of two variables, counting of elements; find min/max of three elements; ordering of three elements by size).

From 6th grade the programming module is directed to transition to programming in script text-based language like Python or JavaScript. The computer modelling module is the last one in the syllabi after the modules Operating system and memory storage media; Word processing; Spreadsheets; Computer graphics; Presentations (including audio); Internet and integration of activities. The programming module is focused on:

- switching from a block programming language to a scripting text programming language with main concepts: text-based programming code, programming environment, input data, output data, variable, assignment operator, conditional operator, loop operator, libraries (turtle, time, random) random numbers, lists with words;
- creating an animation with the tools of a scripting text language with main concept – functions.

In 7th grade the modules are: Computer system and data protection; Word processing; Spreadsheets; Computer modelling; IT Projects development; Audio and video information; Presentations; Internet. The module Computer modelling is focused on two subtopics: Basic data types in a scripting textual programming language with main new concepts numeric (integer and real) data type, strings (text) and second subtopic related to application of loop constructions to solving different tasks based on the learned content in sixth grade.

The new programming module set a lot of challenges.

- Teachers were not ready to implement programming with Python or JavaScript.
- Data types in computer modelling are involved in 7th instead in 6th grade.

- Not enough number of hours for the huge and diversity learning content.
- Lack of enough didactical studies for implementation of script programming at lower secondary school.
- Students have problems with algorithms and own blocks in 5th grade and functions in 6th grade.
- The topic Computer modelling in 5th grade is learned in the middle of the yar, in 6th – at the end of the year, in 7th – in the middle.
- The spiral model leads to forgetting of the learning content in all main topics.
- Choice of appropriate programming environment for script programming.

Despite the many challenges, some of them can be overcome by selecting appropriate training tools and methods. This can be achieved through:

- Experiments with examples prepared in advance and finding new elements in the code.
- Writing in working sheets – technique "paper and pencil".
- Adding additional functionality in "half backed" code.
- Problem based approach.
- Solving tasks from other subjects learned by the students/ cross-curricular approach.
- Use video lessons and demonstrations, provided by the publishers or teachers.
- Finding errors in the code.
- Put the learners in the role of executor of the commands in the code.
- Set tasks related to the content in other topics in the subject CMIT that requires programming.
- Refreshment of the old knowledge and skills before every general topic and lesson.
- Choice of programming environment with easy interface appropriate for beginners.
- Using unplugged activities and role-based games/ exercises.

Some examples for successfully overcoming of the challenges

Once again, teachers and students face challenges such as the abstractness of concepts and the complexity of algorithms. For teachers, the entry level of students is a serious challenge. Omissions made in programming teaching at the initial stage led to difficulties in mastering the new learning material. In addition, the programming module begins in the middle of the school year and students often forget what was learned in the previous programming classes. This can be overcome by making a serious update of the material studied in primary school. It is recommended to be given small projects for story telling or quiz development in block-based environment related to the content learned in previous modules e.g.: about digital devices, about searching for information on the Internet, about the safe use of the Internet, etc. This approach is appropriate also in 6th grade. To keep the knowledge in programming that are achieved in fifth grade the teacher should set periodically small projects similar to the projects in fifth grade that requires story telling or quiz implementation in block programming environment related to previous modules.

To support the understanding and learning of working with own blocks, it is appropriate to use problem-based learning with experiment and finding similarities and differences. The teacher could set a problem situation for introducing new knowledge and students can experiment with the tasks prepared in advance, find new elements in the code. Teachers share about students' difficulties in determining the content of their own block and the input data for the own block. In this case, students can be given tasks to work with paper and pencil with the aim of transformation of code without subroutine to subroutine.

Some examples are available in https://scratch.mit.edu/studios/31790364

Regarding transitions from block-based environment to script-based programming in 6th grade the teachers can:

- choice the programming language and environment. The teachers can choose Python or JavaScript as a programming language. Most of the teachers choose Python. There exists diversity of programming environments, but not all of them are suitable for beginners and for the age of the students. The environments could be online or standalone. Examples of suitable environments for Python programming are standalone environments as Mu (https://codewith.mu/) or Thony (https://thonny.org/). Both of the environments have intuitive interface and allow to be achieved the learning outcomes by the students. Also, online environments as trinket.io (https://trinket.io/) or edublocks (https://edublocks.org/) are very suitable for beginners and allow to be achieved learning goals.
- Use of experiments with code in block-based environment (Scratch) and in script programming language (Python) with the same outputs.
- Rewriting code from Scratch to Python.
- Finding differences and new elements in code.
- Analogy with already used tasks and code.
- Working in worksheets.

In 7th grade one of the specific challenges is related to the real numbers and data type – float. The concept real numbers are part of the 8th grade math curricula. The data type float could be introduced partially with use of examples with decimal fractions.

Some additional approaches include use of tasks with fun or game elements and solving problems step by step, use of tasks related with content from other school subjects or every day live. Examples of cross-curricular tasks that could be implemented in 7th grade according to the learning outcomes: Languages – check the number of signs in text, find number of words in text; Solving tasks from physics, maths etc.; Storytelling about some process learned in other subjects; Maths – drawing figures, surrounding surface of mathematical solids, calculation of percentage, calculation of interest and capital; All subject - creating simple quizzes with questions from other subjects; etc.

The crucial challenge is the readiness of the teachers to teach programming content to students in lower secondary school. Most of the teachers in CMIT in lower secondary school are coming from different subject areas with one-year post-diploma qualification study. Before starting of new curricula in CMIT there were provided courses for teachers – 48 hours in blended mode, but some teachers were not included in the courses and they used self-training to prepare lessons for the students.

Conclusions

The arising challenges in teaching programming in the lower secondary school need of efforts of all stakeholders: teachers, researchers, universities which prepare forthcoming teachers and providers of training courses for teachers; publishers of textbooks; local and national educational authorities. Maybe it is too early for introducing of script programming language in 6th and 7th grade.

It is needed:

- to provide appropriate methodological, teaching and learning resources for teachers and students; to develop of appropriated didactical approaches for teaching informatics at all educational levels.
- before making of changes and introducing of new content it is suitable to be done some research and then apply new methodology in practice.
- to make changes in teachers preparing.

Acknowledgment

The author thanks to the organisers of the conference DIDINFO2025 for the kind invitation and financial support.

References

[1] Tuparov, G., Tuparova, D. (2023). The Ecosystem of Computer Science Education in Bulgarian Primary School – State of the Art. In: Zlateva, T., Tuparov, G. (eds) Computer Science and Education in Computer Science. CSECS 2023. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 514. Springer, Cham. https://doi.org/10.1007/978-3-031-44668-9_26

[2] Sendova E. (2017), Constructionism as an educational philosophy and a culture – a tribute to Seymour Papert, MATHEMATICS AND EDUCATION IN MATHEMATICS, 2017, Proceedings of the Forty-sixth Spring Conference of the Union of Bulgarian Mathematicians, Borovets, April 9–13, 2017, http://www.math.bas.bg/smb/2017_PK/tom_2017/pdf/029-051.pdf

[3] Tuparova, D. Teaching of computer programming in Bulgarian primary school -- challenges and solutions, MIPRO2019, Proceedings of MIPRO 2019, May 20-24, 2019, Opatija Croatia, https://ieeexplore.ieee.org/document/8757040


;
)
REGULAR
CONTRIBUTIONS

„To vědí z domova!“: Vnímání nové informatiky a výuky principů internetu začínajícími učitelkami 1. stupně ZŠ

„They Know It from Home!“: Novice Primary Teachers’ Perceptions of CS Curriculum and Internet Principles

Anna Drobná[1,2], Anna Yaghobová[1], David Šosvald[1], Marek Urban[3], Cyril Brom[1]
[1] KSVI, Matematicko-fyzikální fakulta Univerzity Karlovy, Ke Karlovu 3, 121 16 Praha, ČR
[2] ÚVRV, Pedagogická fakulta Univerzity Karlovy, M. Rettigové 4, 110 00 Praha, ČR
[3] Psychologický ústav Akademie věd České republiky, Pod Vodárenskou věží 1143/4, 182 00 Praha, ČR
drobna@ksvi.mff.cuni.cz, yaghobova@ksvi.mff.cuni.cz, sosvald@ksvi.mff.cuni.cz, urban@praha.psu.cas.cz, brom@ksvi.mff.cuni.cz

EXTENDED ABSTRACT

Computing education in Czech primary schools is undergoing a substantial transformation. Since 2023, the revised national curriculum has mandated the teaching of internet principles from Grade 4 onwards. Yet, little is known about how novice primary school teachers—most of whom have limited technical backgrounds—perceive this curricular shift. This study investigates their awareness of the revised curriculum, their attitudes toward teaching how the internet works, and the rationale they provide either in support of or against including these topics in early education.

Employing a mixed-methods design, the study combines questionnaire data with insights from semi-structured interviews conducted with 60 novice teachers, including both pre-service and early-career in-service educators. Thematic analysis of the interview data reveals that a majority of participants (65%) exhibit only limited understanding of the curriculum changes. Many equate computer science (CS) primarily with digital safety or basic ICT competencies, while only a minority recognize that the new curriculum includes technical content such as internet infrastructure and data transmission.

When asked to assess the importance of specific computing topics, participants consistently prioritized themes related to digital safety—such as secure use of devices and password protection—over technical and conceptual content. Topics such as data flow, network architecture, or the function of servers were frequently perceived as either too complex or insufficiently relevant for younger learners. Notably, most of the participants framed the topic of “how the internet works” through a user-centric lens, focusing on everyday interactions rather than underlying mechanisms.

Overall, teachers’ attitudes towards teaching internet principles ranged from neutral to negative. Concerns included the perceived abstractness of the material, assumptions about pupils’ developmental readiness, and a widespread belief that such knowledge is already acquired informally through family or peer interactions. In addition, several participants expressed a lack of confidence in their own ability to teach technical content, questioning its necessity for developing safe and competent internet users. Nevertheless, the study found that some participants reconsidered their initial attitudes during the interviews as they gained a clearer understanding of what the teaching of internet principles actually entails.

The findings emphasize the need for systematic support in implementing the revised CS curriculum—particularly through strengthening teachers’ conceptual understanding and clarifying how technical knowledge contributes to responsible and safe digital behaviour. Practical implications include the development of targeted teacher training, curriculum-aligned materials, and tools that help educators realistically assess their pupils’ knowledge, rather than overestimating the digital competencies acquired outside of school.

Keywords

Internet, primary school, curriculum revision, teachers, attitudes

ABSTRAKT

Informatika na českých základních školách prochází proměnou. Od roku 2023 jsou principy fungování internetu povinnou součástí výuky již na 1. stupni ZŠ. Není však jasné, jak tuto změnu vnímají začínající vyučující. Studie zkoumá jejich vnímání nové informatiky a postoje k výuce principů fungování internetu. Využívá smíšený výzkumný design kombinující dotazník a polostrukturované rozhovory (N = 60), analyzované tematickou analýzou. Výsledky ukazují, že učitelé mají nízkou informovanost o kurikulární změně a preferují výuku digitální bezpečnosti před technickými aspekty internetu. Jejich postoje k těmto tématům jsou převážně neutrální až negativní. Studie zdůrazňuje potřebu systematické podpory při implementaci nové informatiky do výuky.

Klíčová slova

Internet, 1. stupeň ZŠ, revize RVP, učitelé, postoje

1 ÚVOD

Informatické vzdělávání na českých základních školách prochází zásadní proměnou. Revize Rámcového vzdělávacího programu (RVP), zavedla tzv. novou informatiku[1], jejíž součástí je mimo jiné výuka principů fungování internetu již od 4. ročníku 1. stupně ZŠ [1]. Tato změna reflektuje nejen rostoucí význam digitální gramotnosti, ale také potřebu, aby žáci nejen uměli s technologiemi pracovat, ale zároveň rozuměli jejich základním principům.

Kurikulární reformy jsou obvykle spojovány s příslibem zkvalitnění vzdělávání, avšak jejich úspěšná implementace závisí na pochopení a přijetí ze strany klíčových aktérů, zejména učitelů. V případě nové informatiky sehrávají zásadní roli učitelé 1. stupně ZŠ, kteří mají vyučovat základy informatiky, přestože většina z nich není odborníky na tuto oblast [2]. Jejich postoje k výuce nových témat mohou významně ovlivnit, jakým způsobem bude kurikulární změna realizována v praxi.

Postoje lze definovat jako hodnotící vztah jedince k určitému jevu, tradičně rozlišujeme tři složky postojů: kognitivní (např. znalosti a přesvědčení), afektivní (emoce, pocity, obavy) a behaviorální (chování, implementace) [3]. Tato studie se zaměřuje na **kognitivní** a **afektivní složku** postojů začínajících učitelek k výuce principů fungování internetu, přičemž behaviorální složka (tj. jakým způsobem učitelé tato témata skutečně vyučují) zůstává mimo rozsah této práce.

Výzkum postojů učitelů k informatice se soustředí především na využití technologií ve výuce [4, 5], přičemž roste zájem o jejich postoje k umělé inteligenci [6]. Většina studií však vnímá technologie jako nástroj, nikoli obsah výuky. Zatímco postoje učitelů k práci s internetem jsou relativně dobře zmapované, ochota učit technické aspekty internetu zůstává méně probádaná. Některé studie naznačují, že učitelé jsou k těmto tématům zdrženliví [7], avšak výzkum se zaměřuje především na vyšší stupně vzdělávání. To je pochopitelné, protože výuka principů internetu je v mnoha zemích novým tématem, zejména na 1. stupni ZŠ [8], kde dosud chybí hlubší porozumění učitelským postojům a faktorům, které je ovlivňují.

Cílem této studie je prozkoumat, jak začínající učitelky 1. stupně ZŠ vnímají výuku principů fungování internetu v kontextu nové informatiky, dále jejich informovanost o nové informatice a postoje k výuce principů internetu. V této práci definujeme **principy fungování internetu** v kontextu 1. stupně ZŠ jako základní koncepty, které jsou žákům prezentovány v jednoduché formě, bez hlubší technické složitosti. Konkrétně se zaměřujeme například na **decentralizovanou** povahu internetu, existenci **serverů** a základní mechanismy **přenosu dat**.[2]

---

[1] V tomto textu používáme zavedené slovní spojení *„nová informatika“* pro označení vzdělávací oblasti Informatika v revidovaném českém Rámcovém vzdělávacím programu.

[2] Úroveň znalostí, na něž jsme se během rozhovoru ptali, je k dispozici na následující webové stránce, kterou jsme vytvořili jako podpůrný materiál pro učitele ZŠ. Materiál vznikl na základě potřeb učitelů zjištěných i během tohoto výzkumu. Odkaz na vysvětlení: https://internet4kids.mff.cuni.cz/jak-funguje-internet-vysvetleni/.

2 METODOLOGIE

Tato studie využívá smíšený výzkumný design. Data byla sbírána prostřednictvím:

a) polostrukturovaných individuálních rozhovorů a
b) dotazníku zaměřeného na demografické informace, zkušenosti účastníků s výukou informatiky a sebehodnocení informovanosti o nové informatice.

Polostrukturovaný rozhovor byl zvolen s cílem získat hlubší vhled do chápání principů fungování internetu ze strany účastníků a jejich postojů k výuce těchto témat.

2.1 Účastníci

Do výzkumu bylo zapojeno 60 začínajících učitelů 1. stupně ZŠ (57 žen a 3 muži).[3] Účastníci byli vybráni na základě předem stanovených kritérií, aby spadali do jedné z následujících skupin:

- **Skupina A:** Studenti posledních dvou ročníků oboru učitelství pro 1. stupeň ZŠ, kteří potvrdili svůj záměr věnovat se učitelské profesi po dokončení studia nebo již během výzkumu vyučovali na ZŠ.
- **Skupina B:** Absolventi programu učitelství pro 1. stupeň ZŠ, kteří ukončili studium před dvěma či méně lety a již v době výzkumu vyučovali na ZŠ.

Cílem bylo zajistit vyvážené zastoupení účastníků napříč vysokými školami a pedagogickou praxí. Rovnoměrné zastoupení podle škol se nepodařilo dosáhnout, pravděpodobně kvůli rozdílům v počtu studentů učitelství na jednotlivých školách. Účastníci pocházeli z deseti univerzit, nejvíce z Univerzity Karlovy (n = 17) a nejméně z Masarykovy (n = 2) a Západočeské univerzity (n = 2). Naopak z hlediska pedagogických zkušeností bylo zastoupení vyvážené: vzorek tvořily studentky bez praxe (N = 26) i učitelky s probíhající praxí (n = 34). Většina účastnic byla mladší 25 let (n = 37), 12 bylo ve věku 26–30 let a 11 starších než 30 let. Část absolvovala na vysoké škole didaktiku informatiky pro 1. stupeň ZŠ (n = 16), programování pro děti (n = 14) nebo bezpečnost na internetu (n = 2), zatímco 25 účastnic neabsolvovalo na vysoké škole žádný kurz zaměřený na informatiku.

2.2 Průběh výzkumu

Výzkumné rozhovory probíhaly od listopadu 2022 do března 2023, probíhaly online prostřednictvím platformy Zoom a trvaly 45–60 minut. Sezení probíhala individuálně. Rozhovor vedl vždy jeden ze tří vyškolených výzkumníků, kteří využívali jednotný protokol otázek (ukázka tazatelských otázek v tabulce 1). Na začátku rozhovoru byly účastnice informováni, že cílem výzkumu není testovat jejich znalosti, ale porozumět jejich náhledu a postojům k vybraným tématům nové informatiky.

Tabulka 1: Příklady tazatelských otázek

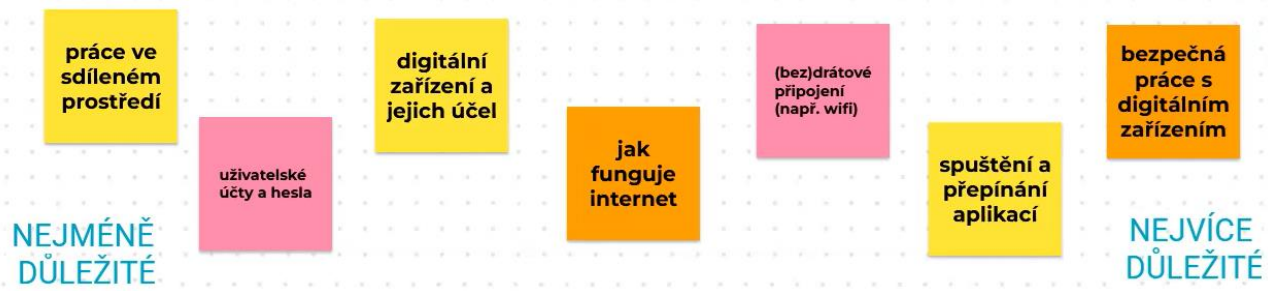
Příklady tazatelských otázek zaměřených na novou informatiku a postoje k výuce principů internetu
Slyšel*a jste o nové informatice nebo revizi RVP? Co se vám pod tím vybaví?
Měl*a jste možnost číst obsah informatiky v RVP?
Považujete některá ze zmíněných témat pro 1. stupeň více důležitá než jiné? Jaká témata to jsou? Proč?
Čeho by se podle vás osobně měla týkat výuka informatiky ve 4. a 5. třídě?

Během rozhovoru byly otázky doprovázeny sdílenými snímky prezentace, které sloužily k lepší orientaci účastnic a také zahrnovali interaktivní otázky (viz obr. 1). Na úvod každého rozhovoru byl

[3] Tento poměr odpovídá dlouhodobým celostátním statistikám, podle nichž tvoří muži přibližně 10 % vyučujících na základních školách v České republice (MŠMT, 2019). Z tohoto důvodu používáme v tomto textu nejčastěji **ženský rod**, mluvíme-li o účastnictvu výzkumu.

kladen důraz na vytvoření příjemné atmosféry, aby se účastnice cítily uvolněně a neváhaly sdílet své názory či myšlenky.

Obrázek 1: Příklad interaktivního snímku prezentace
Zadání: Prohlédněte si učivo z oblasti Digitální technologie nové informatiky z RVP. Srovnejte, prosím, toto učivo podle toho, jak ho sám/sama vnímáte jako důležité.



2.3 Analýza

Nahrávky rozhovorů byly přepsány do textové podoby a následně analyzovány pomocí induktivní tematické analýzy v Atlas.ti 22. Kódování prováděly dvě vyškolené osoby, které nejprve kódovaly společně, aby vytvořily jednotné kódovací schéma. Poté pokračovaly v kódování zbývajících rozhovorů samostatně.

2.4 Etika výzkumu

Studii schválila etická komise Pedagogické fakulty UK. Všechny účastnice podepsaly informovaný souhlas, který je seznámil s cíli, postupy, možnými riziky a přínosy výzkumu. Byly informovány o nahrávání rozhovorů (pouze zvuku a sdílené obrazovky) a o možnosti kdykoli odstoupit. Data byla zpracována anonymně. Účastnice obdržely věcnou odměnu dle výběru (např. powerbanku, flash disk, stolní hru) v hodnotě ~500 Kč.

3 VÝSLEDKY

3.1 Informovanost o „nové informatice“

Účastnice vykazovaly značné rozdíly v úrovni informovanosti o nové informatice. Pouze šest (10 %) jich prokázalo **velmi dobrou informovanost** o této oblasti: byly schopny vlastními slovy popsat některé konkrétní vzdělávací výstupy a obsah spojený s revizí RVP, a dokázaly základně porovnat změny, které revize přináší. Pět z nich v době sběru dat vyučovaly na základní škole. Dvě byly stále studentky, zatímco ostatní čtyři již byly absolventky učitelství, přičemž každá z nich vystudovala jinou univerzitu. Pouze jedna z těchto účastnic vyučovala předmět informatika na 1. st. ZŠ. Přestože prokázaly velmi dobrou informovanost, v rámci sebehodnocení čtyři z nich uvedly, že „spíše nevědí“, co obsahuje nová informatika. Zbývající dvě svou úroveň znalostí hodnotily jako „spíše vím“.

Patnáct účastnic (25 %) bylo zařazeno do skupiny **středně informovaných** o revizi RVP. Tyto účastnice nevykazovaly žádné výrazné společné charakteristiky – pocházejí z různých univerzit, mají rozdílné pedagogické zkušenosti atd. Jejich znalosti o změnách v kurikulu jsou spíše povrchní, nedokážou popsat obsah nové informatiky nebo dopady na výuku informatiky.

Zbylých 39 účastnic (65 %) spadá do kategorie **velmi málo informovaných**. Přestože všechny uvedly, že o revizi RVP slyšely, nedokázaly k nové informatice dodat téměř nic konkrétního.

Kromě analýzy rozhovorů účastnice také odpovídaly na dotazníkovou otázku, zda vědí, co je obsahem nové informatiky. Jejich odpovědi (Obrázek 2) částečně reflektují dříve zmíněné rozdíly v informovanosti. Ne vždy platí shoda mezi tím, jak účastnice hodnotily svou informovanost v

dotazníku, a tím, jak byly následně zařazeny do jednotlivých kategorií na základě rozhovorů. Například jediná účastnice, která v dotazníku označila možnost **„určitě vím“**, byla v rozhovorech zařazena mezi **velmi málo informované**. Zdá se, že některé účastnice podceňovaly nebo naopak přeceňovaly svou informovanost o nové informatice.

Obrázek 2: Odpovědi na otázku *Víte, co je obsahem nové informatiky?*

Novou informatiku si účastnice nejvíce spojovaly s **programováním** a (větším) důrazem na **bezpečnost** na internetu, dále účastnice hojně zmiňovaly vyhledávání a ověřování pravdivosti informací na internetu. Ve vnímání nové informatiky u většiny účastnic jsou patrné velké průniky s jinými oblastmi v RVP ZV – zejména s mediální výchovou a rozvojem digitálních kompetencí. Dalšími často zmiňovanými tématy nové informatiky byla práce v textovém či tabulkovém editoru a tvorba prezentací, k těmto tématům účastnice i mnohdy vyjádřily vlastní (typicky pozitivní) postoj a většinou akcentovaly význam těchto dovedností v budoucím školním i pracovním životě žáka.

> *Vyučující:„Aby si třeba uměli žáci zapnout a vypnout počítač – ano – ale v dnešní době ta doba spěje dopředu a všichni mají doma počítač, takže to mi už tak podstatný zase nepřijde. Ale spíš co děti neumí, to je jak kopírovat, jak pracovat s wordem a jak vlastně tvořit prezentace. Takže tohle je fajn, že je to tam zařazený.“ (žena, 20–25 let, vyučuje na ZŠ, neučí informatiku)*

U některých účastnic bylo jsme zaznamenali zaměňování nové informatiky za *„používání počítačů a tabletů v jiných předmětech“*. Z těchto odpovědí bylo patrné, že vnímají digitální technologie, příp. celou informatiku, jako nástroj pro zvýšení efektivity výuky jiných vyučovacích předmětů.

Zejména učitelky v praxi občas vyjadřovaly také nejistotu nebo obavy z nové informatiky, ať už své osobní, nebo v obavy rámci jejich školy.

> *Vyučující:„Měli jsme jeden nebo dva semestry nějakého počítačového něčeho, ale rozhodně nás to nepřipravovalo na to, že budem učit informatiku, takže jsem se cítila tak jako hozená do vody.“ (žena, 25–30 let, vyučuje na ZŠ, absolventka VŠ r. 2021, učí informatiku)*

> *Vyučující: „U nás se to vlastně začalo řešit v červnu před prázdninami, kdo vlastně bude od září učit novou informatiku. Takže to bylo trošku jako panické, protože se to prostě rozdělilo mezi lidi, kteří vlastně na to neměli moc školení, neměli moc informací, takže spoustu z nich bylo takových docela jako bezradných a moc se jim do toho nechtělo.“ (žena, 30–35 let, vyučuje na ZŠ, neučí informatiku)*

3.2 Hodnocení učiva oblasti Digitální technologie

Účastnice byly požádány, aby seřadily sedm vybraných témat nové informatiky z oblasti Digitální technologie podle důležitosti. Nižší hodnota znamená vyšší důležitost (0 = nejdůležitější, 8 = nejméně důležité). Výsledky jsou znázorněny v Obrázku 3. Jako nejdůležitější považovala většina účastnic (*n* = 41) téma **bezpečná práce s digitálním zařízením**, které dosáhlo průměrného hodnocení 1,60. Druhé nejdůležitější téma pro účastnice představovaly **uživatelské účty a hesla** (2,72). Obě témata souvisejí s digitální bezpečností a prevencí rizik, což během rozhovor zmínily všechny účastnice jako zcela stěžejní témata pro 1. stupeň ZŠ, na něž by měl být kladen největší důraz.

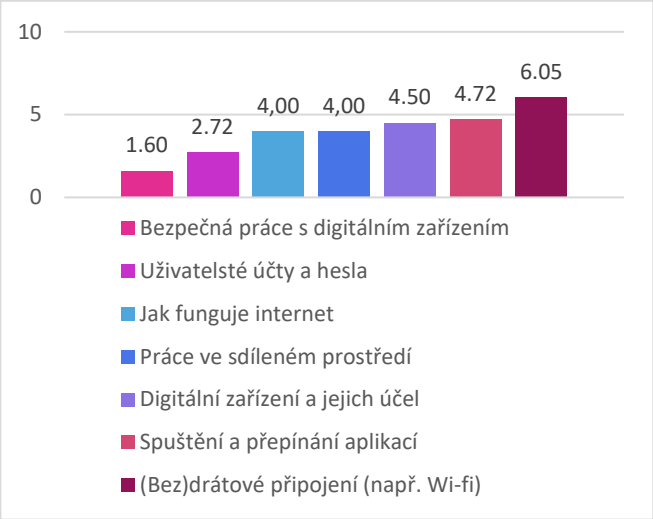
> *Vyučující: „Bezpečí a soukromí na internetu – to si kladu jako já osobně jako prioritu. V posledních jako letech nejsem tak jako sdílný, jako jsem byl (…) někde jsem četl, že když vložíte fotku na Instagram, tak vlastně už to není vaše fotka, ale je to fotka Instagramu, a může si s ní dělat, co chce. Takže z těchto důvodů přestávám být sdílný a bezpečnost na internetu je pro mě téma číslo jedna.“ (muž, 20–25 let, student, nevyučuje na ZŠ)*

Nejméně důležité bylo pro učitelky téma **připojení k internetu**, které dosáhlo průměrného hodnocení 6,05. Podle jejich výpovědí je tato dovednost pro žáky samozřejmostí, protože se s ní setkávají již v

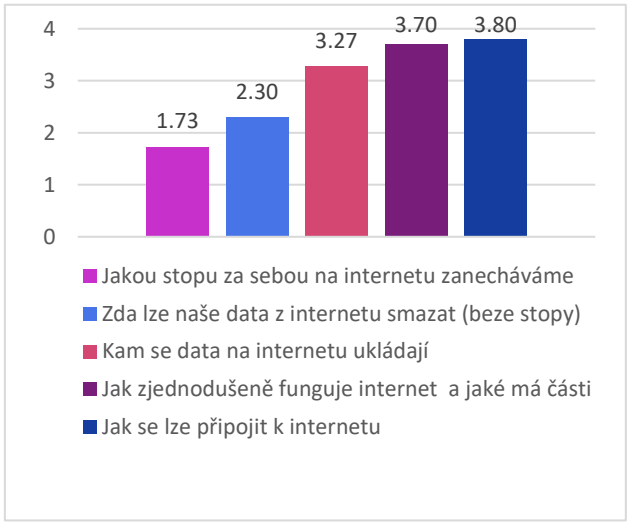
domácím prostředí. Téma vnímají pouze jako dovednost připojit digitální zařízení k internetu, nikoliv porozumění konceptu připojení.

Učivo, které jsme nazvali „**jak funguje internet“**, tj. základní principy fungování internetu, hodnotily účastnice jako středně důležité (4,00). Při podrobnějším doptávání v rozhovorech se ukázalo, že mnoho učitelek chápe toto téma spíše z uživatelského hlediska – jako schopnost pracovat s internetem. To naznačuje, že některé učitelky mohly považovat toto téma za důležité na základě vlastního, odlišného chápání jeho obsahu, než jak bylo původně zamýšleno.

Obr. 3: Hodnocení učiva z RVP ZV



Obr. 4: Hodnocení učiva o fungování internetu



Účastnice výzkumu rovněž seřazovaly pět témat souvisejících s principy fungování internetu podle jejich důležitosti. Výsledky jsou znázorněny na Obrázku 4. Téma **„Jakou stopu za sebou na internetu zanecháváme“** bylo hodnoceno jako nejdůležitější (1,73), následované tématem **„Zda lze naše data z internetu smazat (beze stopy)“** (2,30). I zde je patrné, že účastnice kladou největší důraz na témata související s bezpečností a digitální stopou. Naopak témata zaměřená na technické aspekty internetu – ukládání dat, fungování internetu a způsoby připojení – byla hodnocena jako méně důležitá pro výuku na 1. st. ZŠ. **Připojení k internetu** účastnice opět hodnotily jako nejméně důležité.

3.3 Postoje k výuce principů fungování internetu na 1. stupni ZŠ

Postoje účastnic k výuce principů fungování internetu na 1. stupni ZŠ lze charakterizovat jako převážně neutrální až negativní. V následujících částech této kapitoly popisujeme nejčastější argumenty učitelek pro a proti výuce principů fungování internetu.

3.3.1 Důvody negativního postoje

Složitost či nezajímavost obsahu. Jedním z hlavních argumentů proti výuce principů internetu byla jeho údajná složitost a abstraktnost. Mnoho učitelek (n = 34; 57 %) se domnívá, že žáci 1. stupně ještě nemají dostatečnou schopnost abstraktního myšlení k pochopení těchto konceptů. Některé učitelky zároveň uvedly, že samy principům fungování internetu nerozumí, a proto by pro ně bylo obtížné tyto koncepty vyučovat.

> *Vyučující: „Nevím, jestli to není pro děti zatím třeba **složité**, jakože toto bych zařadila až druhý stupeň – tu část, jak internet funguje.“ (žena, 30–35 let, studentka vyučující na ZŠ, neučí informatiku)*

> *Vyučující: „Děti v tom věku podle mě ještě úplně **nemají takovou tu abstraktní představivost**, tak myslim, že až pro ty starší děti, že by to bylo.“ (žena, 20–25 let, studentka vyučující na ZŠ, neučí inf.)*

Další učitelky vyjadřovaly spíše nezájem žáků o tato témata:

> *Tazatel: „Je podle vás důležité, aby se čtvrťáci nebo páťáci učili o tom, jak funguje internet?“*
> *Vyučující: „Oni už ví, si myslím, jak funguje internet a hlavně je to jakoby až tak moc nezajímá, ono to nějak funguje. Moc je nezajímá jak, hlavně že to funguje. Je to takové, jak to říct, **nezáživné**.“*

Není nezbytné pro uživatelské dovednosti a bezpečnost. Část účastnic (n = 16, 27 %) se domnívala, že znalost fungování internetu není podmínkou pro rozvoj uživatelských dovedností a výuku internetové bezpečnosti.

> *Vyučující: „Upřímně, to, že já třeba **nevím, co je server**, a myslím si, že to ze mě **nedělá horšího uživatele** internetu, protože se snažím chovat bezpečně."*

Přeceňování znalostí žáků. Dalším častým argumentem proti výuce principů internetu bylo přesvědčení učitelek, že děti tyto znalosti získávají mimo školu, nejčastěji v rodině nebo mezi vrstevníky, a škola by tedy tyto oblasti nemusela systematicky rozvíjet (n = 21)**.** Tato představa se objevovala napříč rozhovory, často jako důvod pro menší důraz na výuku fungování internetu.

> *Tazatel: „Je podle vás důležité, aby se čtvrťáci nebo páťáci učili o tom, jak internet funguje?"*
>
> *Vyučující**:** „Já si myslim, že v tom věku už o tom nějaké povědomí mají, skrze rodiče nebo rodinu **už se s tim setkají, takže úplně ty základy asi nejsou nutné v této době**. Každý má ten chytrý telefon, internet a většina lidí to využívá. (…) Takže si jako úplně zásadní věc mi to nepřijde."*
> *(žena, 20–25 let, studentka vyučující na ZŠ, neučí informatiku)*

> *Tazatel: „Když se řekne informatika na 1. stupni, co si pod tím vybavíte?"*
>
> *Vyučující: „No, nějaký základní seznámení dětí s obecně s tim virtuálním světem, aby vůbec třeba věděli, jak takový počítač zapnout, což bych teda řekla, že už **většina dětí ví dokonce líp, než ti učitelé**." (žena, 20–25 let, studentka)*

> *Tazatel: „Čeho by se podle vás osobně měla týkat výuka informatiky ve čtvrté nebo páté třídě?"*
>
> *Vyučující: „(…) také základy **ovládání toho počítače, ale to si myslím, že všechny děti už většinou znají z domu, kolikrát jsou v tom zdatnější než třeba ten učitel**." (žena, 20–25 let, studentka)*

Tento postoj naznačuje tendenci učitelek přeceňovat digitální kompetence žáků nebo zaměňovat uživatelské dovednosti a technické znalosti. Výzkumy přitom ukazují, že ačkoli děti tráví mnoho času online, jejich skutečné porozumění digitálním technologiím je často povrchní [9].

Negativní vztah k technologiím. Vzácně se vyskytly účastnice, které vyjádřily silně negativní postoj k digitálním technologiím obecně. Většinou hovořily o nadmíře užívání technologií žáky ve volném čase.

> *Vyučující: „Já k tomu mám takovej negativní vztah. (…) Já si myslím, že toho jinak mají mimo školu dost, toho počítače a té informatiky vůbec."*

3.3.2 Důvody pozitivního postoje

Navzdory převažujícím negativním postojům se v rozhovorech objevily také argumenty podporující výuku principů fungování internetu (n = 8; 14 %). Nejčastěji zmiňovaly, že děti se s digitálními technologiemi setkávají každý den a měly by rozumět tomu, co se za jejich používáním skrývá.

> *Vyučující: „Cookies, server a tak dál, to by bylo dobrý, aby ty děti věděly, protože třeba s těma cookies se setkávají fakt každej den. Ty děti mají chytrý telefony a měly by vědět, co tam odklikávají."*

3.3.3 Neutrální postoje

Účastnice s neutrálním postojem většinou neuvedly žádné zdůvodnění (n = 5). Některé učitelky se v rozhovorech nechtěly vyjadřovat k výuce principů internetu a odkazovaly se na RVP ZV (n = 2).

> *Vyučující: „Co já si myslim, důležitý není… důležitý je to, co nám jasně přikázalo MŠMT (smích)."*

Z rozhovorů nelze vyvodit jednoznačný postoj každé z účastnic, protože jejich přesvědčení a postoje se během hodinového rozhovoru mnohdy měnily nebo se vyjádřily nejednoznačně k principům jako celku a vyzdvihly jen určité prvky tématu. Celkově však lze říci, že většina účastnic nebyla přesvědčena o důležitosti výuky principů fungování internetu na 1. stupni ZŠ. Přesto však rozhovory ukázaly, že při vysvětlování těchto témat se jejich postoje stávaly pozitivnějšími.

4 DISKUSE A ZÁVĚR

Studie ukazuje výrazné rozdíly v informovanosti začínajících učitelek 1. stupně ZŠ o nové informatice, přičemž většina vykazovala jen minimální informovanost. Některé účastnice svou informovanost podceňovaly, jiné přeceňovaly.

Postoje k výuce principů internetu byly převážně neutrální až negativní. Technické koncepty (např. přenos dat, existence serverů) vnímají jako abstraktní a nepřiměřené pro žáky 1. st. ZŠ. Část učitelek se domnívá, že tyto znalosti děti přirozeně získávají mimo školní vyučování, což však neodpovídá realitě – děti sice technologie využívají, ale jejich porozumění technickým principům bývá nesprávné nebo velmi povrchní [9]. Naopak účastnice pozitivně hodnotí témata jako bezpečné chování na internetu či digitální stopa, což odpovídá předchozím výzkumům. Přitom porozumění bezpečnosti úzce souvisí s pochopením technického fungování internetu.

Postoje učitelek nebyly pevně ukotvené – v rozhovorech je často měnily, zejména poté, co jsme se jich doptávali na konkrétní obsah. To naznačuje, že lepší znalost nové informatiky by jim mohla pomoci zařadit výuku principů fungování internetu do výuky s větší jistotou. Tuto problematiku dále rozvíjíme v připravované studii o vztahu mezi postoji učitelů a jejich znalostmi principů fungování internetu.

Naše zjištění naznačují několik možných implikací pro vzdělávací praxi:

- Je nezbytné pomoci učitelkám lépe porozumět souvislostem mezi znalostí technických principů a bezpečným chováním na internetu. Z našich výsledků vyplývá, že většina učitelek tento vztah nevnímá, což může ovlivnit, jak budou tato témata ve výuce zdůrazňovat nebo zda budou vůbec technické principy vyučovat.
- Bylo by dobré poskytnout učitelkám větší metodickou podporu, například ve formě praktických materiálů nebo školení, které by jim ukázaly, jak efektivně vyučovat principy fungování internetu na 1. st. ZŠ. V současnosti učitelé naleznou podporu k výuce nové informatiky, ale jen vzácně se věnuje principům fungování internetu.[4]
- Bylo by vhodné poskytnout vyučujícím podporu pro rozvoj získávání realistické představy o znalostech a dovednostech žáků. Je stěžejní, aby učitelky nepřeceňovaly jejich znalosti a nepředpokládaly, že si technické aspekty internetu osvojí mimo školu.

Tato studie se zaměřila pouze na začínající učitelky a učitele 1. stupně ZŠ, a její výsledky proto nelze zobecnit na zkušenější pedagogy nebo vyšší stupně vzdělávání. Budoucí výzkum by se měl zaměřit na vývoj postojů s rostoucí praxí a strategie pro efektivní výuku technických témat nové informatiky. Dále je potřeba analyzovat reálnou výuku a obtíže, s nimiž se učitelé při začleňování internetových principů do hodin potýkají.

5 PODĚKOVÁNÍ

Děkujeme za podporu GA UK 484722, GA ČR 22-20771S a SVV 260724. Děkujeme také D. Šťastnému, K. Schubertové, T. Vlčkové, L. Pánkové, všem účastnicím a účastníkům výzkumu.

Při přípravě této práce autoři využili ChatGPT 4o s cílem zlepšit plynulost textu. Po použití nástroje byl obsah textu revidován a autoři přebírají plnou odpovědnost za obsah publikovaného článku.

6 BIBLIOGRAFICKÉ ODKAZY

[1] MŠMT, *Rámcový vzdělávací program pro základní vzdělávání.* Online. Dostupné z: https://www.msmt.cz/file/60264_1_1/, [cit. 1. 2. 2025]

[2] BROM, C., DROBNÁ, A., YAGHOBOVÁ, A., ŠŤASTNÝ, D., ZÁBRODSKÁ, K., & URBAN, M. From Servers to Satellites: Understanding Internet Principles among New Computer Science Teachers. *ACM Transactions on Computing Education*. 2024.

[4] Výjimkou jsou např. výukové materiály Datová Lhota: https://decko.ceskatelevize.cz/datova-lhota

[3] SVENNINGSSON, J., HÖST, G., HULTÉN, M., & HALLSTRÖM, J. Students' attitudes toward technology: exploring the relationship among affective, cognitive and behavioral components of the attitude construct. *International Journal of Technology and Design Education*, *32*(3), 2022. 1531-1551.

[4] CHENG, S. L., & XIE, K. The relations among teacher value beliefs, personal characteristics, and TPACK in intervention and non-intervention settings. *Teaching and Teacher Education*, 2018, 74, 98-113.

[5] SEUFERT, S., GUGGEMOS, J., & SAILER, M. (2021). Technology-related knowledge, skills, and attitudes of pre-and in-service teachers: The current situation and emerging trends. *Computers in Human Behavior*, 115, 106552.

[6] AL DARAYSEH, A. Acceptance of artificial intelligence in teaching science: Science teachers' perspective. *Computers and Education: Artificial Intelligence*, 2023, 4, 100132.

[7] MERTALA, P. The pedagogy of multiliteracies as a code breaker: A suggestion for a transversal approach to computing education in basic education. *British Journal of Educational Technology*, 2021, 52(6), 2227–2241.

[8] ODA, M., NOBORIMOTO, Y., & HORITA, T. International Trends in K-12 Computer Science Curricula through Comparative Analysis: Implications for the Primary Curricula. *International Journal of Computer Science Education in Schools*, 2021, 4(4).

[9] BROM, C., YAGHOBOVÁ, A., DROBNÁ, A., & URBAN, M. 'The internet is in the satellites!': A systematic review of 3–15-year-olds' conceptions about the internet. *Education and Information Technologies*, 2023, 1–30.

Výučba programovania pre nevidiacich žiakov od primárneho vzdelávania po maturitu

Teaching programming blind students from primary education to graduation

Ľudmila Jašková[1], Mária Stankovičová[2]
[1]KDMFI FMFI UK Bratislava, Mlynská dolina F1, 851 04 Bratislava, Slovensko
[2]CPŠ UK Bratislava, Šafárikova nám. 6, 814 99 Bratislava 1,Slovensko
jaskova@fmph.uniba.sk, maria.stankovicova@rec.uniba.sk

EXTENDED ABSTRACT

For more than a decade, we have seen a process of gradual inclusion of computer science as a compulsory subject at all levels of education in many countries. In Slovakia, computer science is a compulsory subject for students aged 8 to 17. Colleagues from our department also participated in the development of the curriculum for this subject. We have a long-standing collaboration with the Support Centre for Students with Special Needs at our university. Within the framework of joint projects, we pay special attention to teaching computer science to blind students, because the way they work with computers is different from the way sighted users work. They interact with the computer using a screen reader. This can only convey text and audio information to them. In addition, blind users do not use a mouse and enter the input using the keyboard. They can therefore only use software environments that work with the screen reader and allow all actions to be performed with the keyboard. Computer skills are essential for them to succeed in education and employment. In addition to building digital literacy, we consider programming and algorithmic problem solving to be a key area of computer science. Examples of successfully employed blind programmers prove that the programming profession is suitable for blind people. However, teaching programming requires the use of a suitable programming environment and also suitable teaching methods. We present a research review in the area of teaching programming to the blind in Chapter 2.

In Chapter 3, we present an overview of programming tools and educational materials for teaching programming to blind students of different ages. We have validated all the environments listed with blind students. Specifically, for primary blind students, we mention the robotic toys Bee-Bot or Blue-Bot in combination with the Tactile Reader board. Next, we mention the physical programming language Code Jumper (also known as Torino). For the lower secondary blind students, we present programming environments developed by our students of applied informatics. There are text-based programming environments for programming music Musik, audio stories Alan and for programming the movement of a virtual object on a sound grid - Coshi and Coshi 2. In the context of upper secondary blind students, we introduce freely available and accessible programming tools and educational materials using the Python language adapted for the blind students.

In Chapter 4, we summarize our findings. The presented programming environments for primary and lower secondary school students assume that teachers can work with blind students in small groups, ideally in special schools. In upper secondary schools, blind students are usually integrated amongst the sighted students. For this target group we have introduced programming environments and educational materials that can be used by all students. It is true that in the adapted educational material we have eliminated tasks focused on programming graphical output. This raises the question of whether blind people will lack the ability to solve such tasks. Finding new technical and educational solutions that will enable them to develop this ability is a challenge for us in the future.

Keywords

Teaching programming, blind students, accessibility, educational materials.

ABSTRAKT

V článku uvádzame prehľad programovacích nástrojov a edukačných materiálov na výučbu programovania pre nevidiacich žiakov rôznych vekových kategórií od 1. ročníka základnej školy až po maturantov. Všetky uvedené prostredia sme overili s nevidiacimi žiakmi. Konkrétne, pre žiakov prvého stupňa ZŠ spomíname robotické hračky Bee-Bot a Blue-Bot v kombinácii s tabuľkou Tactile Reader. Ďalej sa zmienime o fyzickom programovacom jazyku Code Jumper. Pre žiakov druhého stupňa ZŠ sú to textové programovacie prostredia na programovanie hudby Musik, zvukových príbehov Alan a na programovanie pohybu virtuálneho objektu po ozvučenej mriežke – Coshi a Coshi 2. V súvislosti s nevidiacimi žiakmi vyššieho sekundárneho vzdelávania priblížime voľne dostupné a prístupné programovacie nástroje a edukačné materiály využívajúce jazyk Python.

Kľúčové slová

Výučba programovania, nevidiaci žiaci, prístupnosť, edukačné materiály.

1 ÚVOD

Už viac ako desať rokov pozorujeme v mnohých krajinách proces postupného začleňovania informatiky, ako povinného predmetu, na všetky stupne vzdelávania. Viacerí autori uvádzajú argumenty, prečo je dôležité, aby bola informatika povinným predmetom a aký by mal byť jej obsah [1]. Dagiené a kolektív [2] zdôrazňujú, že nie je podstatné učiť produkty vedy a techniky a ich aplikáciu, ale učiť tvorivý proces ich vývoja.

Pri budovaní nášho národného kurikula informatiky spolupracovali aj kolegovia z našej katedry [3], [4]. Máme dlhoročnú spoluprácu s Centrom podpory pre študentov so špecifickými potrebami na našej univerzite. V rámci spoločných projektov venujeme špeciálnu pozornosť výučbe informatiky pre nevidiacich žiakov [5], pretože spôsob ich práce s počítačom je odlišný od spôsobu práce bežných používateľov. Okrem budovania digitálnej gramotnosti považujeme za kľúčovú oblasť informatiky programovanie a riešenie algoritmických úloh. Príklady úspešne zamestnaných nevidiacich programátorov dokazujú, že povolanie programátora je vhodné aj pre nevidiacich. Americká nadácia nevidiacich programátorov eviduje viac ako 130 nevidiacich programátorov, ktorí sú registrovaní ako jej členovia [6]. Sme presvedčení, že čím skôr sa vhodne podchytí záujem nevidiacich žiakov o túto profesiu, tým ľahšie získajú potrebnú kvalifikáciu.

Výučba programovania si však vyžaduje použitie vhodného programovacieho prostredia a vhodných vyučovacích metód. Oboje je v prípade nevidiacich žiakov špecifické. Prehľad niektorých výskumov v oblasti výučby programovania nevidiacich uvádzame v kapitole 2. V kapitole 3 podávame prehľad programovacích nástrojov a edukačných materiálov na výučbu programovania pre nevidiacich žiakov rôznych vekových kategórií od prvého ročníka základnej školy až po maturantov. Všetky prezentované prostredia sme overili s nevidiacimi žiakmi. V kapitole 4 stručne sumarizujeme prezentovanú problematiku.

2 PREHĽAD PROBLEMATIKY

Nevidiaci používatelia pracujú s počítačom pomocou čítača obrazovky. Tento im dokáže sprostredkovať iba textové a zvukové informácie. Okrem toho nevidiaci nepoužívajú myš a vstup zadávajú len pomocou klávesnice. Vhodné sú pre nich iba také softvérové prostredia, ktoré spolupracujú s čítačom obrazovky a umožňujú všetky akcie vykonávať nielen pomocou myši, ale aj pomocou klávesnice.

V ostatných rokoch sa viacerí autori zaoberali prístupnosťou programovacích prostredí pre používateľov so zrakovým postihnutím. Systematický prehľad literatúry o výskume v oblasti prístupnosti programovacích prostredí možno nájsť v publikáciách [7], [8], [9].

Viaceré výskumy sa zameriavali na vývoj a overovanie špeciálnych programovacích prostredí pre nevidiacich žiakov. Napríklad kolektív výskumníkov v Microsoft Research Lab - Cambridge vyvinuli

fyzický programovací jazyk *Torino*, neskôr známy aj ako *Code Jumper* [10] zameraný na programovanie postupnosti zvukov. Bližšie ho opíšeme v časti 3.1.

Obrázok 1: Fyzické programovacie prostredie na programovanie robota DASH [11]

Problematike použitia rôznych dostupných fyzických prostredí na programovanie pohybu robota nevidiacimi žiakmi sa venoval kolektív výskumníkov na univerzite v Lisabone [11]. Jedným zo skúmaných prostredí bol robot *DASH* a tabuľka s *PUZZLETS* blokmi (obrázok 1).

Aj Van der Meulen a kolektív [12] skúmali žiakov so zrakovým postihnutím pri riešení programátorských úloh s robotickými hračkami *Bee-Bot* alebo *Blue-Bot*. Zamerali sa na hlbšie skúmanie ich správania a rôznych úrovní abstrakcie, ktoré žiaci používali pri riešení algoritmických úloh.

Oliveira a kolektív [13] vyvinuli nové programovacie prostredie *GoDonnie*, ktoré zrakovo postihnutí žiaci používali na simuláciu správania robota vo virtuálnom prostredí. Správanie robota sa používateľovi opisovalo prostredníctvom zvukových správ. Príkazy na pohyb a otáčanie robota boli v prirodzenom jazyku[1].

Zaujímavé riešenie ponúka prostredie *Melodic* (obrázok 2) na programovanie melódií, ktoré kombinuje hmatové príkazy a technológiu QR kódov so zvukovým výstupom [14].

Obrázok 2: Hmatové bloky s QR kódmi v programovacom jazyku Melodic [14]

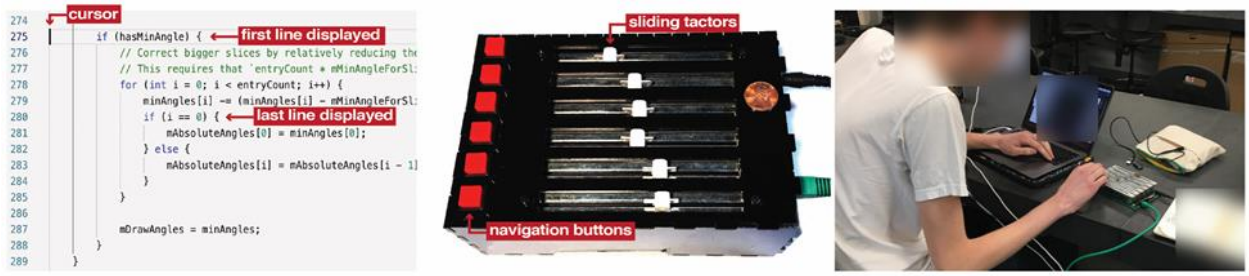
Prístupné programovacie prostredie na programovanie zvukov sa javí ako vhodná alternatíva pre nevidiacich. Za jedno z prvých takýchto prostredí možno považovať prototyp zvukového programovacieho jazyka, ktorý vyvinula skupina programátorov v Chile [15]. Toto prostredie bolo inšpiratívne pre ďalší výskum v oblasti prístupných programovacích prostredí. Vhodným a pomerne jednoduchým prostredím na programovanie hudby pomocou textových príkazov je *SonicPi* [16], ktoré spočiatku nebolo prístupné pre čítač obrazovky, ale aktuálne je už prístupné a plne ovládateľné pomocou klávesnice.

Blokové (resp. kartičkové) jazyky typu *Scratch* sú veľmi populárne pre žiakov základných škôl (ZŠ), pretože odstraňujú problémy súvisiace so zložitosťou syntaxe a sú vhodnou voľbou pre začiatočníkov. Pre nevidiacich žiakov sú však neprístupné. Mountapmbeme a kolektív [17] vytvorili prototyp prístupnej blokovej programovacej knižnice *Accessible Blockly*. Ide o modifikáciu knižnice *Blockly*, ktorá sa používa na vytvorenie väčšiny bežných blokových programovacích prostredí. Knižnica *Accessible Blockly* bola navrhnutá na zlepšenie prístupnosti programovacích prostredí

[1] https://www.youtube.com/watch?v=HE__sAgfNBo&t=45s

založených na blokoch pre osoby so zrakovým postihnutím. Umožňuje im navigovať, vytvárať a editovať kód založený na blokoch len pomocou čítača obrazovky a klávesnice.

Viaceré štúdie sa venovali aj prístupnosti profesionálnych programovacích prostredí. Spomenieme aspoň niektoré z nich. Shaun a kolektív [18] úspešne použili s nevidiacimi žiakmi programovacie prostredie *Ruby*, ktoré sa ukázalo ako plne prístupné. Žiaci v ňom vytvárali programy na analýzu údajov týkajúcich sa fiktívnej ekologickej krízy. Následne písali kód na vytvorenie prístupných hmatových 3D vizualizácií týchto údajov.



Obrázok 3: Použitie hmatového skimmera kódu [19]

Falase a kolektív vedcov zo Stanfordskej univerzity [19] sa vo svojom výskume zamerali na uľahčenie orientácie v štruktúre kódu pre nevidiacich programátorov. Vytvorili hmatový skimmer kódu (Tactile Code Skimmer). Ide o zariadenie, ktoré fyzicky znázorňuje úrovne odsadenia kódu pomocou posuvných potenciometrov. Na obrázku 3 je znázornený čítaný kód (vľavo), jeho zobrazenie na hmatovom skimmeri kódu (uprostred) a programátor (vpravo).

Potluri a kolektív [20] sumarizovali, kategorizovali a opísali početné problémy v oblasti prístupnosti, ktorým čelia vývojári so zrakovým postihnutím pri programovaní v štandardnom používateľskom rozhraní. Vychádzajúc zo svojich zistení vyvinuli doplnok pre prostredie *Visual Studio* zvaný *CodeTalk*, ktorý umožnil vývojárom so zrakovým postihnutím niektoré z týchto problémov prekonať.

3 VÝUČBA PROGRAMOVANIA PRE NEVIDIACICH ŽIAKOV

Tematická oblasť Algoritmické riešenie problémov je pre mnohých učiteľov informatiky oblasťou, ktorú vyučujú s veľkými problémami. Dôvodom je, že mnohí z nich sami nemajú dostatočné programátorské zručnosti, nepoznajú vhodné nástroje pre žiakov a ani osvedčené metodické postupy. Výber vhodného programovacieho prostredia a metodického materiálu preto považujeme za nutnú podmienku úspešného vyučovania tejto tematickej oblasti, obzvlášť v prípade žiakov s ťažkým zrakovým postihnutím. Výskumu v tejto oblasti sa venujeme už viac ako desať rokov. V tejto časti podávame prehľad programovacích prostredí a edukačných materiálov, ktoré sme overili s nevidiacimi žiakmi rôznych vekových kategórií a s ich učiteľmi informatiky. Všetky nami vytvorené prostredia a edukačné materiály sú voľne dostupné na našom webovom portáli *Vyučovanie informatiky nevidiacich*[2].

3.1 Programovanie na primárnom stupni vzdelávania

3.1.1 Robotické hračky

Na výučbu základov algoritmického myslenia v úvodných ročníkoch ZŠ sa nám osvedčilo použitie robotických hračiek *Bee-Bot* alebo *BlueBot* (obrázok 4 v strede) v kombinácií s tabuľkou *Tactile Reader* s kartičkami opatrenými reliéfnymi prvkami (obrázok 4 vpravo).

Tieto robotické hračky majú množstvo vlastností, ktoré ich predurčujú na to, aby sa dali vhodne používať nevidiacimi žiakmi (majú reliéfne ovládacie prvky, ich pohyb sa dá sledovať hmatom a dôležité akcie sú ozvučené). Odporúčame k nim vytvoriť podložku z tvrdého kartónu a lepiacej

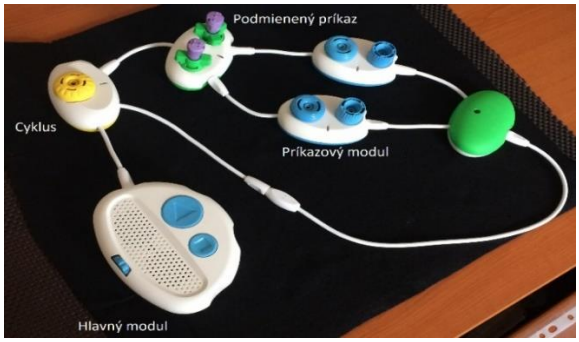
[2] *vin.edu.fmph.uniba.sk*

pásky (obrázok 4 vľavo), aby sa na nej mohli nevidiaci žiaci ľahko orientovať pomocou hmatu. Výskumné overovanie s nevidiacimi žiakmi sme priblížili v [21].

Obrázok 4: Nevidiaci žiak a učiteľ sledujú hmatom pohyb robota Bee-Bot (vľavo), BluBot (v strede), Tactile Reader (vpravo)

3.1.2 Fyzický programovací jazyk

Ďalší programovací nástroj, ktorý nevyžaduje prácu s klávesnicou, je fyzický programovací jazyk *Code Jumper* [8]. Obsahuje hmatovo odlíšiteľné diely (moduly) predstavujúce jednotlivé príkazy, z ktorých môžeme jednoduchým spájaním vytvárať program (obrázok 5). Súvislá postupnosť modulov vytvára vlákno. Tieto vlákna možno pripájať do štyroch vstupov hlavného modulu, pričom na každom vstupe si môžeme zvoliť inú množinu zvukov. Hlavný modul komunikuje s obslužným softvérom v počítači alebo tablete prostredníctvom spojenia Bluetooth. Výskumné overovanie jazyka *Code Jumper* sme podrobnejšie opísali v [22].

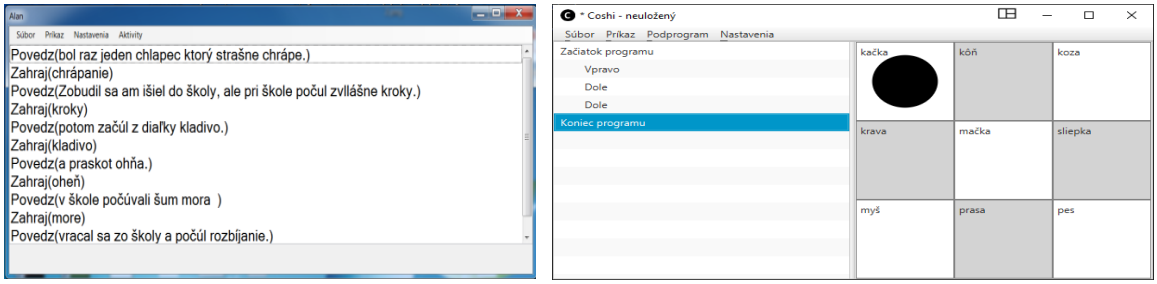


Obrázok 5: Program vytvorený vo fyzickom programovacom jazyku Code Jumper [8]

3.2 Programovanie na nižšom sekundárnom stupni vzdelávania

3.2.1 Blokové programovacie jazyky

Nevidiaci žiaci nižšieho sekundárneho vzdelávania poznajú niektoré klávesové skratky na spustenie programu, dokážu sa dostať do aplikačnej ponuky a vybrať si z nej konkrétnu položku. Môžu používať jednoduché programovacie prostredie, v ktorom si príkazy vyberajú z ponuky a nemusia ich editovať. Takými sú programovacie prostredia *Alan* a *Coshi*. Prostredie *Alan* [23] umožňuje programovať zvukové príbehy (obrázok 6 vľavo) a popri tom rozvíjať tvorivosť a osvojiť si všetky programátorské koncepty požadované osnovami. Výskumnému overovaniu prostredia *Alan* sme sa podrobnejšie venovali v [24].



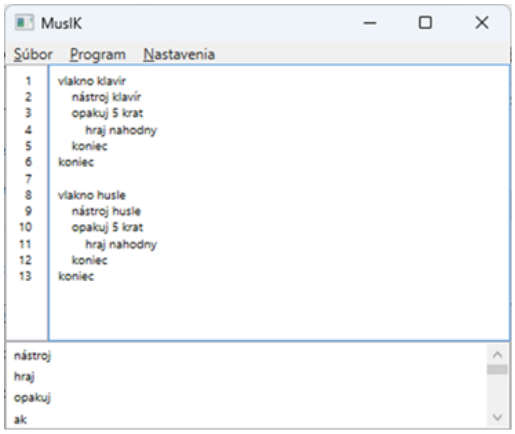
Obrázok 6: Okno aplikácie Alan (vľavo), okno aplikácie Coshi (vpravo)

Pri programovaní sekvencie zvukov si môžu nevidiaci žiaci osvojiť základné programátorské koncepty bez toho, aby boli nútení predstavovať si, čo sa deje na obrazovke počítača. Neskôr je však potrebné u nich rozvíjať aj túto schopnosť. K tomuto účelu slúži programovacie prostredie *Coshi* [24], ktoré umožňuje programovať pohyb robota po ozvučenej štvorcovej sieti na obrazovke počítača (obrázok 6 vpravo). Jeho výskumnému overovaniu sme sa podrobnejšie venovali v [25].

3.2.2 Programovacie jazyky s editorom kódu

Pre nevidiacich žiakov záverečných ročníkov nižšieho sekundárneho vzdelávania sa môže použiť programovacie prostredie vyžadujúce editáciu textu. Takými sú prostredia *Musik* [26] a *Coshi 2* [27]. Na jednoduchšiu tvorbu kódu však tieto prostredia poskytujú možnosť použiť prediktívnu ponuku príkazov. K dispozícii je aj nápoveď so zoznamom a opismi všetkých príkazov. Navigáciu v kóde uľahčuje klávesový príkaz, ktorý umožňuje pohyb po príkazoch v rámci tej istej úrovne vnorenia a získať informácie o úrovni vnorenia. Ak sa po spustení programu vyskytne chyba, aplikácia o tom informuje v špeciálnom paneli (termináli), a keď používateľ stlačí kláves Enter na tomto chybovom hlásení, kurzor sa automaticky nastaví v editore kódu do riadku s chybou.

Aplikácia *Musik* (obrázok 7) umožňuje programovať jednoduché melódie. Melódia môže pozostávať z jedného alebo viacerých vlákien vykonávaných súčasne. Každé vlákno obsahuje postupnosť príkazov prehrávajúcich tóny alebo akordy.



Obrázok 7: Okno aplikácie Musik

Aplikácia *Coshi 2* prevzala mnohé funkcie z aplikácie *Coshi*. Rozdiel je len v editore kódu. *Coshi* používa blokový jazyk (príkazy sa vyberajú z ponuky), naproti tomu v *Coshi 2* používateľ píše kód. Výskumnému overovaniu prostredí *Musik* a *Coshi 2* sme sa venovali v [28].

3.3 Programovanie na vyššom sekundárnom stupni vzdelávania

Na vyššom sekundárnom stupni vzdelávania sa nevidiaci žiaci vzdelávajú prevažne integrovane v bežných školách. Ideálne je preto, aby mohli používať rovnaké programovacie prostredia a edukačné materiály, ako ich intaktní spolužiaci.

3.3.1 Programovacie prostredia

Z hľadiska prístupnosti pre nevidiacich žiakov sme analyzovali niekoľko programovacích prostredí. Žiaľ, prostredia *Idle Python* a *Thonny*, ktoré sa pomerne často používajú na stredných školách, sa ukázali ako nevhodné pre nevidiacich. Nespolupracujú s čítačom obrazovky a nie sú plne ovládateľné pomocou klávesnice. Ako vhodné prostredie sa ukázala pomerne často používaná aplikácia *Visual Studio Code*, nakoľko je plne ovládateľná pomocou klávesnice a spolupracuje s dostupnými čítačmi obrazovky *NVDA* a *JAWS*. Umožňuje programovanie v rôznych programovacích jazykoch, vrátane jazyka Python, ktorý sa už niekoľko rokov používa na stredných školách. Okrem prostredia *Visual Studio Code*, máme dobré skúsenosti aj s online prostredím *Google Colaboratory*[3]. Okrem toho, že

[3] https://colab.google/

je plne ovládateľné pomocou klávesnice a spolupracuje s dostupnými čítačmi obrazovky, je vhodné aj pre používateľov iných operačných systémov ako je Windows. Pre tvorbu jednoduchých programov s výstupom do konzoly je v systéme Windows postačujúca aj *Python konzola*, ktorá je súčasťou čítača obrazovky *NVDA*. Pre iPad v kombinácii s externou klávesnicou máme dobré skúsenosti s programovacím prostredím *Python 3 Ide*, ktoré jedna nevidiaca žiačka gymnázia úspešne používala na hodinách informatiky.

3.3.2 Edukačné materiály

Ako vhodný adaptovateľný edukačný materiál pre výučbu programovania na strednej škole považujeme pracovné listy *abcPython* [29] z nasledujúcich dôvodov.

- Obsahuje dostatok úloh s výstupom do konzoly (terminálu), ktoré umožňujú osvojiť si a rozvíjať osnovami požadované programátorské koncepty.
- K pracovným listom existuje metodický materiál pre učiteľa, ktorý nie je potrebné adaptovať.
- Pôvodné pracovné listy aj metodický materiál [30] boli vytvorené v českom jazyku v rámci projektu PRIM. Boli overené na viacerých školách v ČR rôzneho typu. Prešli recenzným aj vydavateľským konaním.

Adaptácia pracovných listov pre nevidiacich si vyžadovala nasledujúce zmeny.

- Niektoré úlohy, v ktorých sa vyžaduje práca s grafikou, sme zmenili, aby ostala podstata zachovaná, ale výpisy sa vypisovali do terminálu (konzoly), a nie do grafickej plochy. Úlohy, ktoré sa takto zmeniť nedali, sme vynechali.
- Zmenili sme formátovanie textu, aby sa v ňom dalo jednoduchšie orientovať pomocou čítača obrazovky.
- Originálny materiál sa odvolával na prostredie *Idle Python*, v adaptovanom edukačnom materiáli sme sa odvolávali na prostredia prístupné pre nevidiacich.

4 ZÁVER

Predstavili sme riešenia vhodné pre výučbu programovania nevidiacich žiakov rôznych vekových kategórií. Programovacie prostredia a edukačné materiály pre žiakov základnej školy predpokladajú, že učitelia môžu s nevidiacimi žiakmi pracovať v malých skupinách, ideálne v špeciálnych školách. Na stredných školách sú nevidiaci žiaci väčšinou integrovaní medzi intaktných žiakov. Pre túto cieľovú skupinu sme predstavili programovacie prostredia a edukačné materiály, ktoré môžu používať všetci žiaci. Pravda je, že v adaptovanom edukačnom materiáli sme eliminovali úlohy zamerané na programovanie grafického výstupu. Vyvstáva teda otázka, či nebude nevidiacim chýbať schopnosť riešiť takéto úlohy. Hľadanie nových technických a edukačných riešení, ktoré im umožnia rozvíjať túto schopnosť je pre nás výzvou do budúcnosti.

5 POĎAKOVANIE

Tento článok vznikol v rámci výskumov financovaných z grantov agentúry KEGA (014UK-4/2016, 018UK-4/2019, KEGA 010UK-4/2022) a projektu STRIPS (VEGA 1/0407/25). Ďakujeme nevidiacim žiakom základných, stredných a vysokých škôl, ktorí s nami spolupracovali v rámci našich výskumných aktivít a ich učiteľom.

6 BIBLIOGRAFICKÉ ODKAZY

[1] PASSEY, D. Computer science (CS) in the compulsory education curriculum: Implications for future research. In: *Education and Information Technologies*, vol. 22, no. 2, pp. 421-443, Mar 2017. Dostupné z: https://doi.org/10.1007/s10639-016-9475-z, [cit. 4. 2. 2025].

[2] DAGIENÉ, V., HROMKOVIČ, J., LACHER, R. Designing Informatics Curriculum for K-12 Education: From Concepts to Implementations. In: *Informatics in Education*, vol. 20, no. 3, pp. 333–360, 2021.

[3] BLAHO, A., KALAS, I., TOMCSÁNYIOVÁ, M. Experimental curriculum of informatics for 11 years old children. In: *WCCE´95 Liberating the Learner: Proceedings of the sixth IFIP World Conference on Computers in Education*. London: Chapman & Hall, 1995, 829-841. ISBN 0-412-62670-5.

[4] BLAHO, A., SALANCI, L. 2011. Informatics in Primary Schools: Visions, Experiences, and Long-Term Research Prospects. In: KALAS, I. and MITTERMEIR, R., eds. *Informatics in Schools: Contributing to 21st Century Education*, LNCS 7013 Springer, 129 142. ISBN 978-3-642-24721-7.

[5] JAŠKOVÁ, Ľ., KALIAKOVÁ, M. Programming microworlds for visually impaired pupils. In: FUTSCHEK, G., et al., eds. *Constructionism and Creativity 2014*, Proceedings of the 3rd international constructionism conference, Vienna: Östereichische Computer Gesellschaft, 2014.

[6] KONECKI, M. Introductory programming education for visually impaired. In: *IJRET: International Journal of Research in Engineering and Technology*, vol. 03, no: 17, 2014. Dostupné z: https://ijret.org/volumes/2014v03/i29/IJRET20140329012.pdf, [cit. 4. 2. 2025].

[7] DE OLIVEIRA, C. C. *Designing educational programming tools for the blind: mitigating the inequality of coding in schools*, Dissertation, Malmö högskola/Kultur och samhälle, 2017. Dostupné z: https://www.diva-portal.org/smash/get/diva2:1482584/FULLTEXT01.pdf, [cit. 4. 2. 2025].

[8] HADWEN-BENNETT, A., SENTENCE, S., MORRISON, C. Making Programming Accessible to Learners with Visual Impairments: A Literature Review. In: *International Journal of Computer Science Education in Schools*, vol. 2, no. 2, 2018.

[9] OLIVEIRA, et al. Programming teaching with robotic support for people who are visually impaired: a systematic review. In: *Anais Do XXX Simpósio Brasileiro De Informática Na Educação (SBIE 2019)*, 2019. DOI: 10.5753/cbie.sbie.2019.1231

[10] MORRISON, C. et al. Torino: A Tangible Programming Language Inclusive of Children with Visual Disabilities. In: *Human–Computer Interaction*, vol. 35, no. 3, pp. 191–239, 2018. Dostupné z: https://doi.org/10.1080/07370024.2018.1512413, [cit. 4. 2. 2025].

[11] PIRES, A.C., et al. Exploring accessible programming with educators and visually impaired children. In: *Proceedings of the Interaction Design and Children Conference*, 2020.

[12] VAN DER MEULEN, A., et al. Observing the computational concept of abstraction in blind and low vision learners using the Bee-bot and Blue-bot. In: *Computer Science Education*, pp. 1–23, 2023. Dostupné z: https://doi.org/10.1080/08993408.2023.2272232, [cit. 4. 2. 2025].

[13] OLIVEIRA, et al. GoDonnie: A Robot Programming Language to Improve Orientation and Mobility Skills in People Who are Visually Impaired. In: *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '19)*. New York: Association for Computing, pp. 679–681, 2019. Dostupné z: https://doi.org/10.1145/3308561.3354599, [cit. 4. 2. 2025].

[14] COSTA, R., ARAÚJO, C., HENRIQUES, P. R. Melodic - Teaching Computational Thinking to Visually Impaired Kids. In: *Second International Computer Programming Education Conference (ICPEC 2021)*, *Open Access Series in Informatics (OASIcs)*, vol. 91, pp. 4:1-4:14, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, Dostupné z: https://doi.org/10.4230/OASIcs.ICPEC.2021.4, [cit. 4. 2. 2025].

[15] SÁNCHEZ, J., et al. Blind learners programming through audio. In: *CHI'05 extended abstracts on Human factors in computing systems*, ACM, 2005, pp. 1769-1772.

[16] AARON, S. *Code music with Sonic Pi*. Dostupné z: https://www.raspberrypi.org/magpi-issues/Essentials_Sonic_Pi-v1.pdf, [cit. 4. 2. 2025].

[17] MOUNTAPMBEME, A., OKAFOR, O., LUDI, S. Accessible Blockly: An Accessible Block-Based Programming Library for People with Visual Impairments. In: *Proceedings of the 24th*

International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '22). Association for Computing Machinery, New York, NY, USA, Article 19, 2022, pp. 1–15. Dostupné z: https://doi.org/10.1145/3517428.3544806, [cit. 4. 2. 2025].

[18] KANE, S. K., BIGHAM, J. P. Tracking@ stemxcomet: teaching programming to blind students via 3D printing, crisis management, and twitter. In: *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 2014, pp. 247–252.

[19] FALASE, O., SIU, A. F., FOLLMER, S. Tactile Code Skimmer: A Tool to Help Blind Programmers Feel the Structure of Code. In: *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '19)*. Association for Computing Machinery, New York, NY, USA, 2019, pp. 536–538. Dostupné z: https://doi.org/10.1145/3308561.3354616, [cit. 4. 2. 2025].

[20] POTLURI, et al. CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, Paper 618, 2018, pp. 1–11. Dostupné z: https://doi.org/10.1145/3173574.3174192, [cit. 4. 2. 2025].

[21] JAŠKOVÁ, Ľ. Ako začať s informatikou na ZŠ pre nevidiacich žiakov? In: TRAJTEĽ, Ľ., eds. *Didinfo 2013*, 2013, FPV UMB - Banská Bystrica, s. 104-109, ISBN 978-80-557-0527-4.

[22] KOVÁČOVÁ, N., JAŠKOVÁ, Ľ., KARASOVÁ, M. Dva programovacie jazyky priateľské k nevidiacim žiakom nižšieho sekundárneho vzdelávania In: HORVÁTHOVÁ, D., eds. *Didinfo 2019*, Banská Bystrica : Univerzita Mateja Bela, 2019. ISBN 978-80-557-1533-9.

[23] KOVÁČ, Matúš. *Programovacie prostredie pre zrakovo postihnutých žiakov ZŠ*, bakalárska práca, Bratislava: FMFI UK, 2016.

[24] KOVÁČ, Michal. *Programovacie prostredie na ovládanie pohybu virtuálneho objektu prístupné pre nevidiacich žiakov sekundárneho vzdelávania*, bakalárska práca, Bratislava: FMFI UK, 2019.

[25] BUDINSKÁ, L., JAŠKOVÁ, Ľ., KARASOVÁ, M., KOVÁČ, M., MIKOVÁ, K. Mikrosvet na výučbu programovania priateľský k nevidiacim študentom sekundárneho vzdelávania In: DRÁBKOVÁ, J., BERKI, J., eds. *Didinfo 2020*. Liberec, 2020. ISBN978-80-7494-532-8, ISSN 2454-051X. Dostupné z: http://www.didinfo.net/images/DidInfo/files/Didinfo_2020.pdf, [cit. 4. 2. 2025].

[26] ŠVORC, J. *Edukačné prostredie na programovanie hudby prístupné pre nevidiacich žiakov sekundárneho vzdelávania*, diplomová práca, Bratislava: FMFI UK, 2024.

[27] SOJKA, A. R. *Programovanie virtuálneho objektu prístupné pre nevidiacich žiakov sekundárneho vzdelávania*, diplomová práca, Bratislava: FMFI UK, 2024.

[28] JAŠKOVÁ, Ľ., SOJKA, A.R., ŠVORC, J. Two programming environments for the development of algorithmic thinking and creativity of lower secondary blind students. In: *ICERI 2024 Proceedings*, pp. 1655-1663. ISBN: 978-84-09-63010-3.

[29] BLAHO, A., SALANCI, Ľ. *abcPython Pracovné listy pre Python*. Dostupné z: https://abcpython.input.sk/, [cit. 4. 2. 2025].

[30] BLAHO, A., SALANCI, Ľ., ŠIMANDL, V. *Programování v jazyce Python pro střední školy.* PedF JU České Budějovice. Dostupné z: https://imysleni.cz/ucebnice/zaklady-programovani-v-jazyce-python-pro-stredni-skoly, [cit. 4. 2. 2025].

Modul pre kolaboratívnu tvorbu kvízových otázok
Module for collaborative creation of quiz questions

Katarína Krupková
Univerzita Komenského, FMFI
Mlynská dolina
84248 Bratislava
Slovensko
krupkova6@uniba.sk

Zuzana Kubincová
Univerzita Komenského, FMFI
Mlynská dolina
84248 Bratislava
Slovensko
kubincova@fmph.uniba.sk

EXTENDED ABSTRACT

Quizzes are an effective learning tool that promotes active and self-regulated learning. Research shows that regular testing can help students identify gaps in their understanding of the material and improve long-term knowledge retention. In addition, quizzes are motivating because they provide immediate feedback and thus encourage further learning. They promote critical thinking as students must analyze the questions and choose the correct answers. These benefits of quizzes are even more pronounced when students directly create the quizzes. Incorporating such activities into the classroom promotes effective and engaging learning. In our paper, we present the Quiz module, which is a part of the Courses LMS developed at FMFI UK. The module was developed for collaborative creation of quiz questions by students, and we also present the first results of its preliminary testing.

The purpose of the module is to be used as an educational tool – for instance, a teacher might assign students to create quiz questions on a particular topic. The module currently supports creation of multiple-choice questions with predefined answers. Feedback can be given by commenting on a question. It is possible to edit an existing question, by its author or the teacher of the course, while preserving the previous versions. Teachers can also "approve" questions, which are then added to a question bank used to generate quizzes.

These quizzes are intended to serve two main purposes – students can use them for self-testing, while teachers can assign quizzes to students as assessment. In the current implementation, these generated quizzes serve as a self-testing tool. Students can enter their answers and receive immediate feedback. The result is only informative and is not recorded.

The Quiz module has undergone three rounds of testing. The first two rounds were carried out with student teachers, while the third round was conducted at a secondary school. The participants were given brief instructions on working with the module and were asked to carry out several tasks within the module. After the test, they filled out a questionnaire containing open-ended questions about their experience while working with the module, potential issues they encountered and ideas for improvements. The features of the module were added and modified between rounds of testing according to the module's specification, as well as the received feedback.

The results of the testing show that students appreciate the collaborative character of the module, as well as its role in fostering active learning. Consistent with findings of several previous studies, participants consider the creation of quiz questions as a useful tool for self-study. Furthermore, the participating student teachers saw its potential for their future teaching. The main challenge is the time-consuming nature of creating questions and distractors. The testing also revealed several areas for improvement of the module. We plan to continue research on the module's use, as we believe that engaging students in quiz creation can enhance both the effectiveness and appeal of the learning process.

Keywords

Quiz, testing, quiz questions, collaborative question creation, learning activities.

ABSTRAKT

Kvízy sú efektívnym vzdelávacím nástrojom, ktorý podporuje aktívne a samoregulované učenie sa. Výskumy ukazujú, že pravidelné testovanie môže študentom pomôcť identifikovať medzery v porozumení učiva a lepšie a dlhodobejše si zapamätať novonadobudnuté vedomosti. Okrem toho kvízy pôsobia motivujúco, pretože poskytujú okamžitú spätnú väzbu a tým povzbudzujú k ďalšiemu učeniu. Podporujú kritické myslenie, keďže študenti musia analyzovať otázky a vybrať správne odpovede. Tieto výhody kvízov sú ešte výraznejšie, ak študenti kvízy priamo tvoria. Zaradenie takýchto aktivít do vyučovacieho procesu podporuje efektívne a zaujímavé učenie sa. V našom príspevku prezentujeme modul Quiz, ktorý je súčasťou LMS Courses vyvinutého na FMFI UK. Modul bol vyvinutý na kolaboratívnu tvorbu kvízových otázok študentmi a v článku prezentujeme aj prvé výsledky jeho predbežného testovania.

Kľúčové slová

Kvíz, testovanie, kvízové otázky, kolaboratívna tvorba otázok, vzdelávacie aktivity.

1 ÚVOD

Testy a kvízy sa často používajú ako nástroje formatívneho hodnotenia na rôznych stupňoch vzdelávania a v rôznych vzdelávacích oblastiach [1-3]. Ako ukázal Kwan vo svojej porovnávacej štúdii [4], kvízy sú dokonca efektívnejšími nástrojmi formatívneho hodnotenia, ako niektoré iné bežne používané nástroje.

Ich pravidelné využívanie vo vyučovaní môže žiakom pomôcť lepšie pochopiť vyučovanú látku a zlepšiť ich študijné výsledky [5]. Cohen ukázal pozitívny vzťah medzi prístupom študentov k učeniu a ich skóre v online kvízoch. Po absolvovaní série kvízov študenti preukázali výrazné zlepšenia v čase, ktorý strávili riešením online kvízu a zlepšovali sa aj ich výsledky [1]. Podobne, silnú koreláciu, resp. kauzálnu súvislosť medzi absolvovaním formatívnych kvízov a zlepšeným výkonom na záverečnej skúške naznačujú aj výsledky ďalších výskumov [2, 6]. Štúdia vykonaná medzi študentmi prvého ročníka na holandskej vysokej škole [2] ukázala, že študenti, ktorí absolvovali všetky moduly formatívneho kvízu, mali výrazne vyššie skóre na záverečnej skúške. Platilo to pre študentov, ktorí aj predtým dosahovali dobré výsledky, ale ešte výraznejšie zlepšenie vykazovali študenti, ktorí predtým dosahovali len slabé výsledky.

Zhang [7] dokonca tvrdí, že formatívne kvízy môžu nielen zlepšiť výkon študentov na skúške, ale aj predpovedať ich študijné výsledky. K podobným záverom dospel aj Kibble, ktorý v štúdii o samotestovaní ako formatívnom hodnotení [8] poukázal na to, že výkon študentov pri ich prvom pokuse v online kvízoch bol prediktívny pre ich výkon na záverečných skúškach.

Výsledky priebežných kvízov možno zaradiť aj ako súčasť výsledného hodnotenia, avšak aj v takomto prípade väčšinou tvoria len zlomok celkového hodnotenia, preto ich študenti chápu skôr ako pútavú, prípadne aj zábavnú aktivitu, ktorá im pomáha lepšie sa sústrediť na preberanú látku [9, 10]. Viaceré štúdie ukazujú, že študenti vysoko hodnotia používanie online kvízov ako súčasť výučby a považujú ich za užitočný vzdelávací nástroj [10, 11]. Kvízy ako nástroj online formatívneho hodnotenia môžu zvýšiť motiváciu a zapojenie študentov do edukačných aktivít [3, 5], čo sa prejavuje ešte viac, ak ide o adaptívne kvízy, ktoré upravujú obťažnosť na základe predchádzajúceho výkonu študentov [9].

Spomínané zlepšenie výkonu študentov vplyvom kvízov môže však súvisieť aj s tým, že priebežné zisťovanie úrovne vedomostí ich samotných motivuje k aktívnemu učeniu sa a lepšiemu organizovaniu svojho vzdelávania, čo v konečnom dôsledku pomáha zlepšiť ich výsledky [12-15].

Surip vo svojej prehľadovej štúdii [9] spomína aj výskumy poukazujúce na to, že pokiaľ sú kvízy len nepovinnou súčasťou kurzu a ničím neprispievajú do celkového hodnotenia, študenti ich nevyužívajú

a keď je za ne ponúknutá odmena vo forme čo i len minimálneho počtu bodov, tak študenti kvízy využívajú a nakoniec z nich na záverečnej skúške profitujú.

Iná štúdia [6] dospela k záveru, že väčšina študentov má tendenciu skúšať samotestovacie kvízy až tesne pred záverečnou skúškou, čo podľa nášho názoru znižuje ich efektívnosť ako vzdelávacieho nástroja. Ak sú kvízy zadávané pravidelne, študenti sú nútení držať krok s výučbou a postupne sa učiť preberanú látku počas celého kurzu [11, 12, 15], čo prispieva k jej lepšiemu a dlhodobejšiemu zapamätaniu [13, 14, 16].

Používanie kvízov má množstvo výhod nielen pre študentov, ale aj pre učiteľov. Kvízy využívané na formatívne hodnotenie poskytujú učiteľom informácie o výsledkoch a postojoch študentov [1]. Patria medzi významné metódy na rýchlu a účinnú identifikáciu slabých a silných stránok študentov a hodnotenie priebežného pokroku študentov, čo sú informácie, ktoré môžu pomôcť zlepšiť výučbu a dosiahnuť ciele a zámery študijného programu [17].

Trochu iným smerom sa uberajú výskumy, ktoré skúmajú výhody kvízov vytvorených študentmi vo vzdelávacom prostredí. Štúdie ukázali, že tento prístup môže viesť k vyššej angažovanosti študentov, môže im pomôcť hlbšie pochopiť preberanú látku a aj zlepšiť ich študijné výsledky [18, 19]. Tvorba kvízov pre spolužiakov je formou aktívneho učenia sa, ktoré podporuje u študentov pochopenie cieľov predmetu, sebareflexiu a zvyšuje ich schopnosť samoregulovaného učenia sa [19]. Autori viacerých štúdií prezentujú aj digitálne platformy, ktoré umožňujú študentom vytvárať kvízy, absolvovať ich a diskutovať o nich [18-21] a uvádzajú pozitívne reakcie študentov, ale aj také objektívne zistenia, ako je napr. štatisticky významné zlepšenie študentov pri riešení úloh.

V snahe motivovať a aktivizovať našich študentov sme sa rozhodli do našej výučby zapojiť aj aktivity, v ktorých budú študenti kolaboratívne tvoriť kvízové otázky k danému kurzu. V tomto príspevku prezentujeme prvú verziu modulu Quiz, ktorý bol za týmto účelom vyvinutý ako súčasť nášho vzdelávacieho prostredia courses.matfyz.sk. Prvé testovanie modulu so študentmi pomohlo odladiť niektoré jeho nedostatky a doplniť požadovanú funkcionalitu, ale tiež ukázalo ich záujem o aktivity tohto typu.

2 MODUL QUIZ

2.1 Návrh modulu

Požiadavky na modul boli definované na základe literatúry a diskusie s vývojovým tímom prostredia Courses.

Hlavným účelom modulu je tvorba, kontrola a úprava kvízových otázok rôzneho typu, ako aj generovanie kvízov z dostupných otázok. Modul by sa mal využívať ako učebný nástroj –učiteľ môže napríklad zadať študentom úlohu vytvoriť v systéme kvízovú otázku. K vytvorenej otázke by mali členovia kurzu môcť poskytnúť spätnú väzbu a tým prispieť k jej zrozumiteľnosti a vyššej kvalite. Autor otázky by otázku mal vedieť upraviť alebo aj vymazať. Učiteľ má mať tiež možnosť pridávať otázky, upravovať ich alebo vymazať. Pri úprave sa pôvodná verzia otázky zachová v histórii.

Ak je otázka kvalitná, učiteľ má mať možnosť otázku schváliť. To znamená, že otázka je zaradená do banky otázok, z ktorých sa zostavujú kvízy. Generovanie kvízov má mať dva účely – študenti môžu generovať kvízy na samotestovanie, kým učitelia môžu študentom priradiť kvíz, ktorý bude súčasťou hodnotenia.

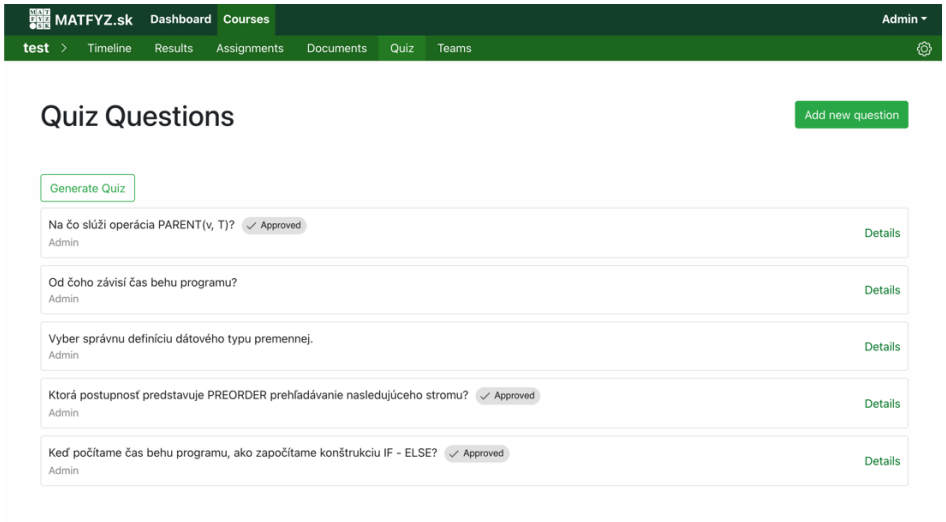
2.2 Funkcie a implementácia

2.2.1 Použité technológie

Modul Quiz, ako aj celé prostredie, je webová aplikácia. Používateľské prostredie využíva knižnicu React[1] založenú na JavaScripte. Jej výhodou je možnosť vytvárať dynamické stránky a viacnásobné použiteľné komponenty. Na globálne manažovanie stavu webovej aplikácie a komunikáciu s backendom je využitá knižnica Redux[2] a jej nadstavba Redux Toolkit[3].

2.2.2 Súčasti modulu

Domovská stránka. V úvodnom zobrazení je zoznam všetkých otázok, ktoré účastníci kurzu pridali. Je možné pozrieť detail otázky, generovať kvíz alebo pridať novú otázku.



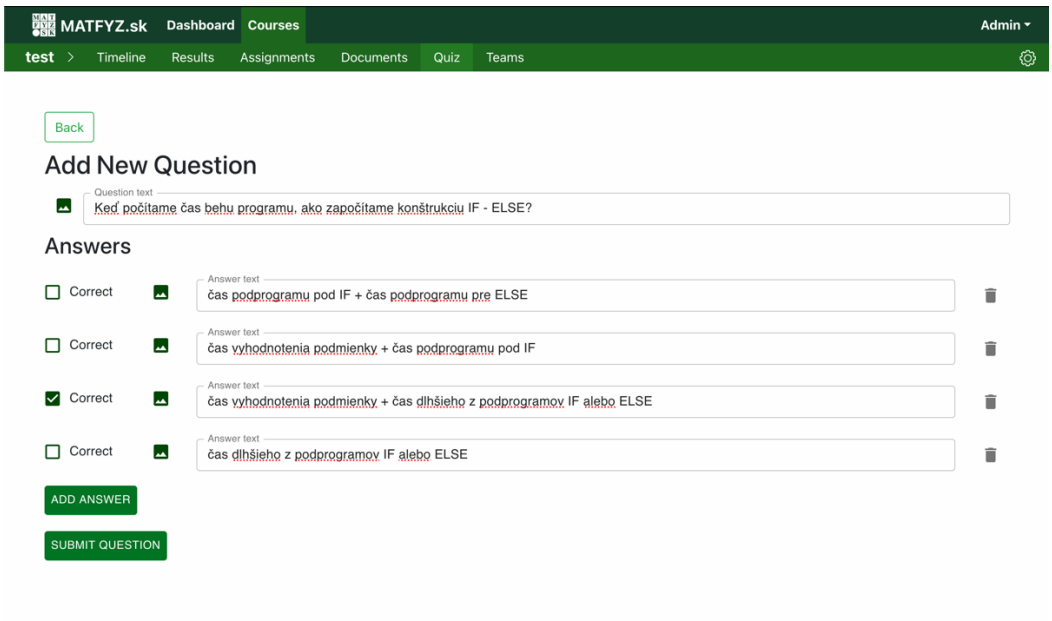
Obrázok 1: Domovská stránka modulu Quiz

Pridávanie otázok. Ktorýkoľvek člen kurzu môže pridať kvízovú otázku. Momentálne je možné pridávať otázky s preddefinovanými možnosťami odpovede, ktorých môže byť ľubovoľné množstvo. Aspoň jedna z odpovedí, ale aj viaceré, môžu byť označené ako správne. K možnostiam odpovedí môžu byť pridané aj obrázky. Pred odoslaním na server je správnosť vstupov a veľkosť priložených súborov validovaná. V prípade chýb sa zobrazí správa.

[1] React. © 2024 Meta Platforms, Inc. Dostupné z: https://react.dev

[2] Redux. © 2025 Dan Abramov and the Redux documentation authors. Dostupné z: https://redux.js.org

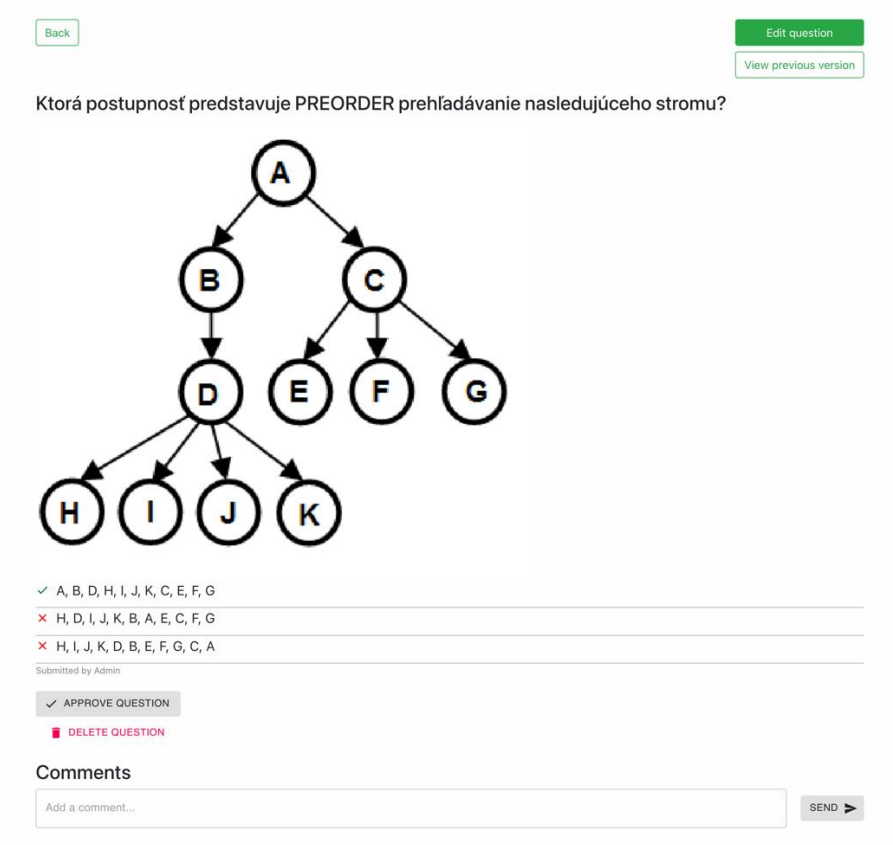
[3] Redux Toolkit. © 2025 Dan Abramov and the Redux documentation authors. Dostupné z: https://redux-toolkit.js.org



Obrázok 2: Pridávanie otázky

Detail otázky. Toto zobrazenie slúži na diskusiu a spätnú väzbu o otázke. Používateľ vidí text otázky, možnosti odpovedí a aj to, ktoré odpovede autor označil ako správne.

Pod otázkou je sekcia komentárov, ktoré môže pridávať ktorýkoľvek člen kurzu. Pri komentári je viditeľné autorovo meno, prípadne prezývka, ak ju má nastavenú.

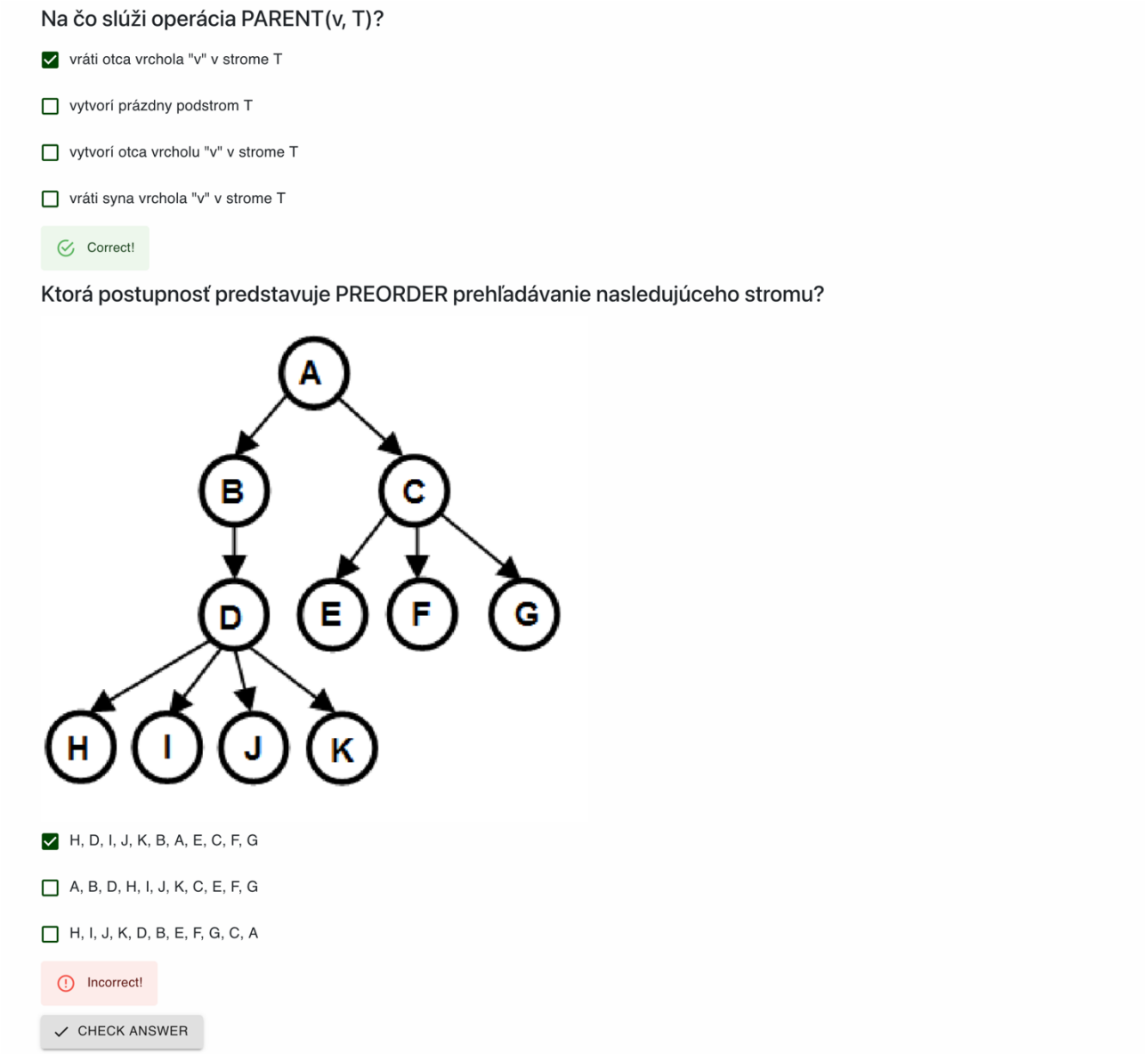


Obrázok 3: Detail otázky z pohľadu učiteľa

Ďalej môže na tejto stránke autor otázky alebo učiteľ otázku upraviť. Ktorýkoľvek člen kurzu si môže pozrieť predchádzajúcu verziu otázky, ak existuje. Učiteľ môže otázku schváliť.

Editovanie otázky. Úprava otázky využíva rovnaký pohľad ako pridávanie. Používateľ môže vykonať ľubovoľné zmeny, ale upravená otázka musí taktiež prejsť validáciou.

Generovanie kvízu. Pri generovaní kvízu sa náhodne vyberie a zobrazí 10 položiek z banky schválených otázok. Kvízy momentálne fungujú na úrovni samotestovania. Po vyplnení môže používateľ skontrolovať svoje odpovede.



Obrázok 4: Vygenerovaný kvíz po kontrole odpovedí

3 TESTOVANIE MODULU

Modul Quiz prešiel tromi kolami testovania. V prvých dvoch kolách išlo o testovanie so študentmi vysokej školy, kým účastníci tretieho kola boli žiaci strednej školy. Účastníci dostali inštrukcie na prácu s testovacou verziou modulu a na záver vyplnili dotazník. Medzi jednotlivými kolami sa funkcionalita modulu menila a dopĺňala podľa návrhu modulu a získanej spätnej väzby.

3.1 Pilotné testovanie

Cieľom prvého kola testovania bolo identifikovať a odstrániť chyby v module a overiť, či je používateľské rozhranie dostatočne intuitívne. Účastníci nemali žiadne predošlé skúsenosti s modulom.

Účastníci testovania pracovali pod jedným zdieľaným účtom. Ich úlohou bolo vytvoriť jednu otázku, upraviť ju a potom upraviť jednu otázku od iného autora. Následne vypĺňali dotazník s otvorenými otázkami o ich skúsenosti s prácou v module, prípadných chybách, na ktoré narazili, alebo odporúčaniach na možné vylepšenia. Keďže účastníci boli študenti učiteľstva, zaradili sme aj otázky na potenciálne využitie v ich budúcej učiteľskej praxi.

Pri testovaní sa nezistili žiadne závažné chyby a všetci zúčastnení boli pri práci s modulom úspešní. Napriek tomu sme získali cennú spätnú väzbu, ktorú sme zapracovali v ďalšom vývoji. Na otázku potenciálneho využívania z perspektívy učiteľa sme získali jednoznačne kladné odpovede. Z pohľadu študentov boli odpovede zmiešané. Ako hlavné negatívum spomenuli účastníci časovú náročnosť vymýšľania otázok.

3.2 Testovanie na vysokej škole

Druhé kolo testovania prebehlo s desiatimi študentmi učiteľstva informatiky v rámci kurzu Algoritmy a údajové štruktúry. Účastníci dostali inštrukcie, aby si vopred pripravili kvízovú otázku o prebratom učive.

Študenti mali tento raz samostatné účty a otázky bolo možné komentovať. Súčasťou testovacieho procesu bolo pridanie vlastnej otázky, pridanie spätnej väzby na cudziu otázku formou komentára, a následná úprava svojej otázky podľa získaných komentárov. Názory účastníkov sme zbierali formou dotazníkov.

V tomto kole testovania sa nenašli žiadne chyby a účastníci považovali prostredie za intuitívne. Najčastejším spomínaným nedostatkom modulu bola chýbajúca možnosť pridania obrázkov k otázkam a odpovediam.

Aj v tomto kole sa účastníci zhodli, že by takúto aktivitu využívali z pohľadu učiteľa. Ako výhody spomínali podporovanie aktívneho učenia a kolaborácie. Jeden účastník však spomenul riziko, že si študenti pri tvorbe distraktorov omylom asociujú s otázkou nesprávnu odpoveď.

3.3 Testovanie na strednej škole

Tretieho kola testovania sa zúčastnila najväčšia skupina 116 žiakov strednej odbornej školy, ktorých zameranie je programovanie digitálnych technológií. Vzhľadom na to bolo súčasťou inštrukcií aj hľadanie chýb a zraniteľností systému a návrh vylepšení. Z dotazníka sme pre toto kolo vypustili otázky o použiteľnosti modulu z perspektívy učiteľa.

Väčšina účastníkov považovala modul za užitočný a uviedla, že by takúto aktivitu využila pri vlastnom vzdelávaní. Vyzdvihli aj kolaboratívny aspekt modulu. Viacerí žiaci uviedli, že podobnú metódu už využívajú v papierovej forme alebo na inej platforme.

Približne štvrtina žiakov sa vyjadrila, že by takýto modul nevyužila. Spomínali viaceré dôvody: niektorým metóda nevyhovovala vzhľadom na ich štýl učenia sa, iní vnímali nedostatok funkcionality, ktorá by zvýšila užitočnosť modulu. Ďalšou opakujúcou sa odpoveďou bolo to, že vytváranie otázok je časovo náročné a trávenie času vymýšľaním distraktorov nie je efektívne.

Od žiakov sme získali veľké množstvo nápadov na vylepšenie. Taktiež sme odhalili chyby v module, napríklad v spracovaní konkrétnych reťazcov textu alebo v správaní tlačidiel, keď bolo možné rýchlym klikaním otázku pridať viackrát po sebe.

Viaceré návrhy od žiakov boli po testovaní zapracované do modulu. Pridali sme dialógové okno, ktoré sa zobrazí, ak používateľ chce opustiť stránku vytvárania otázky bez uloženia. Taktiež sme ku komentárom pridali čas ich pridania a možnosť upraviť a vymazať ich. Ďalšie hodnotné pripomienky boli odložené do budúceho vývoja modulu.

4 ZÁVER

Prezentovaný modul Quiz je vzdelávací nástroj zameraný na kolaboratívnu tvorbu kvízových otázok a generovanie kvízov.

V súlade s viacerými publikovanými štúdiami [10, 11, 18] výsledky testovania modulu ukázali, že študenti pozitívne vnímajú jeho kolaboratívny charakter a podporu aktívneho učenia. Podobne, ako uviedli Stelovsky a Posselt [18, 19], aj naši študenti považovali tvorbu kvízových otázok za užitočnú pre ich samoštúdium. Navyše, študenti učiteľských študijných programov videli v module potenciál na využitie v pedagogickej praxi. Hlavnou výzvou je časová náročnosť pridávania otázok a vymýšľania odpovedí. Testovanie modulu zároveň odhalilo viaceré oblasti na vylepšenie používateľského zážitku a funkcionality.

V budúcnosti by sme modul radi rozšírili o ďalšie typy otázok, napríklad otvorené otázky, esejové otázky, či priraďovanie dvojíc. Plánujeme tiež pridať možnosť vyhľadávať otázky a filtrovať ich podľa autora, obťažnosti alebo témy. Ďalším užitočným rozšírením je podpora formátovania textu otázok a odpovedí, ktorá by umožnila prácu s matematickými symbolmi a formátovanie kódu. Do testovania modulu by sme chceli zahrnúť aj učiteľov, ktorých perspektíva môže byť nápomocná pre ďalší vývoj a smerovanie modulu.

Avšak už aj teraz – bez plánovaných rozšírení – sa chystáme využiť modul Quiz vo vyučovaní, pretože si myslíme, že integrácia aktivít, v ktorých študenti tvoria kvízy, môže byť cestou k pútavejšiemu a efektívnejšiemu vzdelávaciemu prostrediu.

5 POĎAKOVANIE

Autorky ďakujú za podporu projektom VEGA 1/0621/22 a APVV-20-0353.

6 BIBLIOGRAFICKÉ ODKAZY

[1] COHEN, Donita, and Irit SASSON. Online quizzes in a virtual learning environment as a tool for formative assessment. *JOTSE* 6.3, 2016, pp.188-208

[2] ZIJLSTRA, Sjirk-Jan J., et al. Formative Quizzing and Learning Performance in Dutch First-Year Higher Education Students. *Computer Assisted Assessment. Research into E-Assessment*: 18th International Conference, CAA 2015, Zeist, The Netherlands, June 22–23, 2015. Proceedings 18. Springer International Publishing.

[3] GIKANDI, Joyce Wangui, Donna MORROW, and Niki E. DAVIS. Online formative assessment in higher education: A review of the literature. *Computers & education* 57.4, 2011, pp. 2333-2351.

[4] KWAN, Felix. Formative Assessment: The One-Minute Paper vs. the Daily Quiz. *Journal of Instructional Pedagogies* 5, 2011.

[5] CHUNLIANG Yang, Liang LUO, Miguel VADILLO, Rongjun YU, and David SHANKS. Testing (quizzing) boosts classroom learning: A systematic and meta-analytic review. *Psychological Bulletin*, 147, 03, 2021.

[6] ARAVINTHAN, Vasantha, and Thiru ARAVINTHAN. Effectiveness of self-assessment quizzes as a learning tool. *Proceedings of Engineering Education Conference* (EE 2010), 2010.

[7] ZHANG, Niu, and Charles NR HENDERSON. Can formative quizzes predict or improve summative exam performance?. *Journal of Chiropractic Education* 29.1, 2015, pp. 16-21.

[8] KIBBLE, Jonathan. Use of unsupervised online quizzes as formative assessment in a medical physiology course: effects of incentives on student participation and performance. *Advances in Physiology Education* 31.3, 2007, pp. 253-260.

[9] SURIP, Noor, Zulkifli SOM, Muthukumar PALANISAMY, and Mazliza MOHAMAD. Ideas for Designing Better Quizzes: A Literature Review and Suggestion. *International Journal of Academic Research in Progressive Education and Development*, 10, 08, 2021.

[10] SANDI, Sandi, and Dangin DANGIN. Benefits and Barriers on Quizizz Application as Formative Assessment Tool in Vocabulary Teaching: Students Side. *EDULIA: English Education, Linguistic and Art Journal* 4.2, 2024, pp. 166-177.

[11] SALAS-MORERA, Lorenzo, Antonio ARAUZO-AZOFRA, and Laura GARCÍA-HERNÁNDEZ. Analysis of online quizzes as a teaching and assessment tool. *JOTSE: Journal of technology and science education* 2.1, 2012, pp. 39-45.

[12] MCDANIEL, M., P. AGARWAL, B. HUELSER, K. MCDERMOTT, and H. ROEDIGER. Test-enhanced learning in a middle school science classroom: The effects of quiz frequency and placement. *Journal of Educational Psychology*, 103, 2011, pp. 399–414.

[13] ROEDIGER III, Henry L., and Jeffrey D. KARPICKE. The power of testing memory: Basic research and implications for educational practice. *Perspectives on psychological science* 1, 3, 2006, pp. 181-210.

[14] KARPICKE, Jeffrey D., and Henry L. ROEDIGER III. Repeated retrieval during learning is the key to long-term retention. *Journal of memory and language* 57, 2, 2007, pp. 151-162.

[15] ROEDIGER III, Henry L., Adam L. PUTNAM, and Megan A. SMITH. Chapter one - ten benefits of testing and their applications to educational practice. *Psychology of Learning and Motivation*, 55, Academic Press, 2011, pp.1–36.

[16] ROEDIGER III, Henry L., and Jeffrey D. KARPICKE. Test-enhanced learning: Taking memory tests improves long-term retention. *Psychological science* 17, 3, 2006, pp. 249-255.

[17] STUFFLEBEAM, D. L. *The CIPP Model for program evaluation*. Boston: Kluwer: Nijhoff Publishing, 2003

[18] STELOVSKY, Jan, Michael-Brian OGAWA, Branden OGATA, and Umida STELOVSKA. Constructive learning by teaching: Flip-flop, peer evaluation, and agile tooltip: Making and taking peer quizzes synchronized with lecture screencasts. *Designing for the user experience in learning systems*, 2018, pp. 177-200.

[19] POSSELT, Charlotte, Charlotte FLYGER, and Niels KJAERSGAARD. Strengthening students' learning outcomes through students working with quizzes. *Towards a new future in engineering education, new scenarios that european alliances of tech universities open up*, Universitat Politècnica de Catalunya, 2022, pp. 1454-1461.

[20] DI PIERRO, Massimo, and Peter HASTINGS. Social Quizzes with Scuiz. *Recent Advances in Information Systems and Technologies*, 2, 5, Springer International Publishing, 2017, pp. 975-982.

[21] THURZO, Andrej, M. MAKOVNÍK, J. LYSÝ, L. VALKOVIČ, and V. JAVORKA. Medical e-learning and testing system powered by crowdsourcing and based on social networks designs. *Mefanet report* 03, 2010, p. 126.

Význam rozpoznávania miskoncepcií vo vyučovaní programovania

The Importance of Identifying Misconceptions in Teaching Programming

Gabriela Lovászová
Fakulta prírodných vied a informatiky UKF v Nitre
Tr. A. Hlinku 1
949 01 Nitra
Slovensko
glovaszova@ukf.sk

Viera Michaličková
Fakulta prírodných vied a informatiky UKF v Nitre
Tr. A. Hlinku 1
949 01 Nitra
Slovensko
vmichalickova@ukf.sk

EXTENDED ABSTRACT

The article focuses on the issue of errors in the cognitive process of students in introductory programming courses, specifically how these errors can lead to incorrect, incomplete, or distorted understanding and the creation of misconceptions. The article aims to:

- define the conceptual framework related to researching misconceptions in programming,
- outline methods for identifying misconceptions and investigating their causes,
- provide examples of programming misconceptions in practice.

In the article, concepts are defined as "categories of objects, activities or states of being that have certain properties." These are formed through abstraction. Thinking processes involve creating relationships between concepts, forming structures, and developing "conceptions" as systems of ideas based on concepts. Errors and distortions in this process can lead to "alternative, naive (preconceptions), incorrect (misconceptions) ideas" that are not aligned with generally accepted scientific knowledge.

The article outlines different research methods for studying misconceptions depending on the research goals. Exploratory research aims to discover what misconceptions students have, it is typically qualitative in nature. Verification research methods tests the presence of known misconceptions on a larger sample, with results easily evaluated quantitatively. Several specific methods for identifying misconceptions are clarified: think-aloud protocols, conceptual mapping, semi-structured interviews, analysis of errors in source code, and conceptual tests.

Several illustrative examples of misconceptions observed in introductory programming courses, particularly among future and practicing Informatics teachers, are presented. The observed misconceptions are categorized based on the type of programming knowledge involved: syntactic knowledge (understanding syntax), conceptual knowledge (understanding algorithms and logic), and strategic knowledge (understanding problem-solving strategies).

A significant point raised is that misconceptions can be transferred from teacher to student. This underscores the importance of well-prepared teachers who possess a strong understanding of programming concepts and effective teaching methodologies. Our future research focuses on working with Informatics teachers to address their misconceptions, acknowledging that improving teacher competence is crucial for improving student learning in programming.

Keywords

Teaching programming, error, preconception, misconception, alternative conception.

ABSTRAKT

Článok je zameraný na problematiku chýb v poznávacom procese žiakov a študentov v úvodných kurzoch programovania. Tie môžu viesť k nesprávnemu, neúplnému alebo skreslenému porozumeniu a vytváraniu mylných predstáv – miskoncepcií. Cieľom je vymedziť pojmový aparát súvisiaci so skúmaním miskoncepcií v programovaní a načrtnúť metódy odhaľovania miskoncepcií a skúmania príčin ich vzniku. V článku sú vymedzené pojmy koncept, koncepcia, prekoncepcia, miskoncepcia, alternatívna koncepcia a objasnené niektoré metódy na identifikovanie miskoncepcií vo všeobecnosti. Uvedených je niekoľko príkladov identifikovania miskoncepcií v kontexte vyučovania programovania. V závere je načrtnutý význam odbornej a metodickej spôsobilosti učiteľov informatiky pracovať s chybami a miskoncepciami žiakov a viesť ich k budovaniu kvalitných syntaktických, konceptuálnych a strategických vedomostí z programovania.

Kľúčové slová

Vyučovanie programovania, chyba, prekoncepcia, miskoncepcia, alternatívna koncepcia.

1 ÚVOD

Pri analýze a hodnotení žiackych výkonov je učiteľ konfrontovaný s rôznou kvalitou výkonov. Výsledky vzdelávacieho procesu totiž nezávisia len na vonkajších faktoroch, ako je osoba učiteľa, obsah, formy, metodika vyučovania, vzdelávacie prostredie, ktoré sú pre skupinu žiakov v triede rovnaké, ale predovšetkým na individuálnych charakteristikách žiakov, ako sú aktuálny stav poznania a poznávacích schopností, osobnosť žiaka, psychické rozpoloženie, životná situácia a podobne. Základom poznávacieho procesu je totiž myslenie ako „zložitý psychický proces formovania a nového zobrazenia dostupných informácií“ [1, s. 157], ktorý prebieha vo vedomí jednotlivca a manipuluje s poznatkami v jeho poznatkovom systéme. Dôležitou charakteristikou procesu myslenia jednotlivca je aj to, že smeruje k riešeniu problémov, ktoré ho zaujali.

Poznávací proces je teda vnútorným psychickým procesom, ktorý je jedinečný pre každého žiaka. Jeho podstatu vo všeobecnosti zhrnul Hejný ako trojicu „motivácia – skúsenosti – poznanie“ [2]. Tieto tézy sú základom konštruktivistických prístupov vo vyučovaní, ktoré vyučovanie chápu ako podporovanie vnútorného poznávacieho procesu žiaka vytváraním podnetného prostredia, sprostredkovaním získavania skúseností, poskytovaním informácií a metodickej podpory.

V tomto príspevku sa zameriavame na jednu z príčin nízkych žiackych výkonov na vyučovaní, ktorou sú chyby v poznávacom procese. Tie môžu viesť k nesprávnemu, neúplnému alebo skreslenému porozumeniu a vytváraniu mylných predstáv – miskoncepcií. Predmetom nášho záujmu je oblasť programovania, ktoré patrí medzi náročnejšie témy vo vyučovaní informatiky, kde je výskyt nízkej kvality žiackych výkonov častý. Naším cieľom je vymedziť pojmový aparát súvisiaci so skúmaním miskoncepcií v programovaní a načrtnúť metódy odhaľovania miskoncepcií a skúmania príčin ich vzniku.

2 CHYBY A MISKONCEPCIE V PROGRAMOVANÍ

Pojmy (*koncepty*) sú „kategórie objektov, činností alebo stavov bytia, ktoré majú určité vlastnosti“ [1]. Vznikajú abstrakciou – vyčlenením podstaty od detailov v rámci nejakého kontextu. V procese myslenia sa vytvárajú medzi konceptmi vzťahy, usporadúvajú sa do štruktúr, vznikajú *koncepcie* ako systémy usporiadania myšlienok založených na konceptoch. V literatúre sa tiež môžeme stretnúť s pojmom *mentálny model* ako kognitívna reprezentácia vonkajšieho sveta v mysli jednotlivca (predstava) [3, 4].

V procese myslenia – či už ide o vytváranie mentálneho modelu sveta alebo tvorbu pojmov a koncepcií – môže dochádzať k chybám, skresleniam a vzniku takých predstáv, ktoré v rôznej miere

nie sú v súlade s aktuálnym všeobecne akceptovaným vedeckým poznaním. Takéto predstavy sa zvyknú označovať ako *alternatívne*, naivné (*prekoncepcie*), nesprávne (*miskoncepcie*) [4, 5].

Štrauch a Hanč [6] spresňujú pojem miskoncepcia ako „výsledok uvažovania zahrňujúci nesprávne mentálne modely, pričom takýto výsledok sa vyskytuje v danej populácii študentov s významnými pravdepodobnosťami.“ Posúvajú tým jeho chápanie z individuálneho výskytu javu u jednotlivca, na kategóriu javov, ktoré nie sú ojedinelým prípadom, ale vyskytujú sa u rôznych jednotlivcov opakovane.

Gavala a Lukáč [7] upozorňujú na potrebu odlíšenia pojmov *chyba* a *miskoncepcia*. Kým *bežné chyby* môžu byť epizodické, spôsobené napríklad nepozornosťou, nesústredenosťou alebo nedostatkom relevantných informácií, a majú jednoznačné vonkajšie prejavy, miskoncepcie sú *chyby v konceptuálnom porozumení*, ktoré sa ťažko odhaľujú, vyznačujú sa stabilitou, a preto je náročnejšie ich odstraňovať.

Výskyt chýb a miskoncepcií v myslení žiakov je prirodzený jav, nemožno ho považovať za niečo negatívne. Chyba je užitočnou formou spätnej väzby pre žiaka aj pre učiteľa a dobrou príležitosťou na rozmýšľanie do hĺbky. Problémom však je, ak miskoncepcia zostáva neodhalená a hlboko ukotvená. Odhaľovanie a publikovanie známych miskoncepcií v programovaní, napr. [8], je preto pre pedagogickú prax mimoriadne užitočné. Učiteľ tak získava sprostredkované skúsenosti o existencii možných chýb v konceptuálnom porozumení a môže sa na ne vopred metodicky pripraviť.

3 METÓDY IDENTIFIKOVANIA MISKONCEPCIÍ

V edukačnom výskume zameranom na miskoncepcie sa v závislosti od cieľov výskumu uplatňujú rôzne metódy zberu a analýzy dát. V rámci *exploračného výskumu* sa snažíme zistiť, aké miskoncepcie študenti majú. Výskyt niektorých sa dá očakávať, objaviť však môžeme aj úplne nové. Takýto výskum má typicky kvalitatívny charakter. V prípade *verifikačného výskumu* testujeme prítomnosť už známych miskoncepcií na väčšej vzorke študentov, jeho výsledky sa ľahko vyhodnocujú kvantitatívne.

K exploračným metódam na objavovanie miskoncepcií patria najmä:

Think-aloud protokoly: Študent dostane programátorskú úlohu a je požiadaný, aby nahlas hovoril o tom, čo robí a prečo. Výskumník ho neopravuje ani nenavádza na konkrétne riešenie alebo pracovný postup, iba pozoruje a nahráva jeho slovný komentár. Získané dáta však môžu poskytnúť komplexný a detailný vhľad do procesu myslenia a problémov študenta pri tvorbe riešenia, ktorý nie je možný spoľahlivo rekonštruovať na základe analýzy odovzdaného výsledku práce [9].

Konceptuálne mapovanie je vizuálna technika, ktorá pomáha zobraziť, ako študenti chápu vzťahy medzi rôznymi programátorskými konceptmi. Študent dostane kľúčové pojmy a má ich usporiadať do schémy s prepojeniami. Vzťahy medzi pojmami sú označené popismi (napr. „skladá sa z“, „dedí z“, „používa“ , "zavolá" a pod.). Pri tvorbe takejto pojmovej mapy si môžu študenti sami uvedomiť svoje miskoncepcie alebo nedostatočné porozumenie. Porovnaním máp môžeme získať prehľad o mentálnych modeloch viacerých študentov [10].

Semištruktúrované rozhovory: Skúsený výskumník vedie rozhovor o programátorskej úlohe so študentom podľa série pripravených otázok, ale s možnosťou prispôsobiť smerovanie rozhovoru odpovediam študenta s cieľom porozumieť jeho mysleniu a identifikovať príčiny jeho miskoncepcií [11].

Analýza chýb v zdrojovom kóde je metóda, ktorá pomáha hľadať opakujúce sa vzorce nesprávneho myslenia v autentických riešeniach študentov. Táto metóda poskytuje reálne dáta, ktoré ukazujú časté chyby a miskoncepcie. Spracovanie vytvorených programov môže byť manuálne, ale aj automatizované alebo poloautomatizované, a teda aplikovateľné na veľké vzorky študentov [12]. Túto metódu možno preto efektívne využívať aj pri verifikačnom výskume. Jej nevýhodou je však fakt, že takto odhaľujeme len symptómy miskoncepcií, ale nie ich príčinu. Nie je možné s istotou rozlíšiť situáciu, keď študenti rozumejú konceptu, ale urobia syntaktickú chybu.

Ku kvantitatívnym metódam na zisťovanie výskytu konkrétnych miskoncepcií patria okrem analýzy chýb v programoch aj špeciálne pripravené *konceptuálne testy* navrhnuté na základe zistení výskumu o tom, ako študenti rozmýšľajú a kde robia chyby. Distraktory v testoch sú založené práve na bežných miskoncepciách. Konceptuálne testy poskytujú objektívny pohľad na časté problémy u študentov, umožňujú tiež sledovať zmeny v porozumení pred a po výučbe, učitelia môžu prispôsobiť učebné materiály alebo metodiku na základe zistených miskoncepcií. Sú efektívnym prostriedkom formatívneho hodnotenia.

Vychádzajúc zo štúdia výskumných prác, ale aj dlhoročných vlastných skúseností v oblasti vyučovania programovania, môžeme potvrdiť, že je to práve *verbalizácia postupu riešenia*, ktorá poskytuje najviac informácií o spozorovanej alebo potenciálnej miskoncepcii, súvislostiach jej vzniku, príp. o dôvodoch jej pretrvávania. Okrem slovného komentára vlastného alebo vzorového riešenia sa nám v rámci didaktiky programovania v učiteľskom štúdiu osvedčujú aj také úlohy, v ktorých študenti tvoria písomné poznámky k vybranej téme z programovania v podobe vhodnej pre žiakov základnej alebo strednej školy alebo formulujú úlohy pre žiakov, ktoré sú analogické ku vzorovým. Pri väčších programátorských projektoch študenti vystupujú s prezentáciou svojho riešenia pred spolužiakmi alebo odovzdávajú štruktúrovanú písomnú dokumentáciu, v ktorej okrem opisu riešenia reflektujú svoju skúsenosť s programovaním a uvažujú o prípadných nedostatkoch svojho riešenia. Nekvalitná ústna alebo písomná prezentácia riešenia obyčajne spoľahlivo ukazujú na nesprávne myšlienkové a pracovné postupy autora alebo prezrádza povrchné porozumenie niektorým konceptom.

4 PRÍKLADY DIDAKTICKÝCH SITUÁCIÍ

Zdrojom niekoľkých nasledujúcich príkladov didaktických situácií, v ktorých boli identifikované naivné koncepcie, alternatívne koncepcie alebo miskoncepcie sú študentské riešenia úloh v úvodnom kurze programovania. Študenti sú budúci učitelia alebo učitelia z praxe, ktorí navštevujú rozširujúce štúdium informatiky na získanie kvalifikácie vyučovať informatiku. Chyby, ktoré sa v programoch vyskytli, boli diskutované v rozhovore s učiteľom. Študenti slovne opisovali svoje riešenia a odpovedali na otázky učiteľa.

4.1 Vstup z klávesnice

Didaktická situácia: Študent má vyriešiť úlohu, v riešení ktorej je potrebné na začiatku načítať zo vstupu nepárne prirodzené číslo.

Program:

```
n = input('Zadaj nepárne číslo: ')
```

Výpočet:

```
Zadaj nepárne číslo: 21
```

Na otázku učiteľa *„Akú hodnotu má premenná n?“* študent odpovedá jednoznačne *„21“*.

Učiteľ vyzve študenta, aby svoju odpoveď overil. Tu môžu nastať dve situácie: Študent použije na overenie kontrolný výpis v editore programov a spustí program znova, alebo zobrazí hodnotu premennej v príkazovom riadku. Pri použití kontrolného výpisu

Program:

```
n = input('Zadaj nepárne číslo: ')
print(n)
```

Výpočet:

```
Zadaj nepárne číslo: 21
21
```

výpis potvrdí jeho odpoveď, hoci výsledkom nie je číslo, ale reťazec. Z tejto skúsenosti môže vzniknúť miskoncepcia, že funkcia `input` môže vracať číselné hodnoty. Vedieme preto študentov

k experimentovaniu a zobrazovaniu hodnôt v príkazovom riadku bez spracovania funkciou `print`, pokiaľ je to možné.

```
>>>n
'21'
```

Zobrazenie hodnoty v príkazovom riadku odhalí chybu – `n` je znakový reťazec, nie číslo. Študent vytvorí z hodnoty premennej `n` číslo pomocou funkcie `int`:

```
>>>int(n)
21
```

a upraví program takto:

```
n = input('Zadaj nepárne číslo: ')
int(n)
```

Toto riešenie odhaľuje miskoncepciu, že hodnota premennej `n` sa mení volaním funkcie `int(n)`. Jej pôvod vidíme v ďalšej miskoncepcii, že študenti často nepokladajú priradenie = za samostatný príkaz, ale len za nejakú formu zápisu. Potvrdením tejto miskoncepcie je odpoveď na otázku *„Akým príkazom sa premennej n priradila vstupná hodnota?“*. Študent odpovedá *„input()“*.

4.2 Mená premenných a hodnoty

V tomto príklade uvádzame kostru programu:

```
smer = input("Zadaj smer šípky (l/p): ")
if smer == l:
    ## vykreslenie šípky vľavo
else:
    ## vykreslenie šípky vpravo
```

Výpočet vygeneruje chybu:

```
NameError: name 'l' is not defined
```

V tomto prípade sa na prvý pohľad zdá, že nejde o miskoncepciu, ale chybu z nepozornosti – študent zabudol na apostrofy alebo úvodzovky pri porovnávaní hodnoty premennej `smer` s hodnotou `'l'`. Učiteľ preto položí otázku *„Akého typu je premenná smer?“* Študent správne odpovie *„string“* a upraví podmienku na

```
smer == str(l)
```

Takéto riešenie však odhalí nedostatočné pochopenie rozdielu medzi premennou a hodnotou, čo vedie k mylnej predstave o riešení chyby. Problém nie je v type hodnoty, ale v neexistencii odkazovaného mena. Jednou z príčin tejto miskoncepcie je nedostatok skúseností s interpretovaním správy o chybe, ale tiež chyba v metodickom postupe učiteľa, ktorý svojou otázkou upriamil pozornosť študenta nesprávnym smerom. Lepšia otázka: *„Je l premenná alebo konkrétna hodnota?“*

Nedostatok skúseností s interpretovaním správy o chybe demonštruje aj nasledujúci príklad, v ktorom študent prvé slovo zo zoznamu `slova` priraďuje do premennej s menom `1`:

```
1 = slova[0]
```

Výsledkom je správa o chybe:

```
SyntaxError: cannot assign to literal here. Maybe you meant '==' instead of '='?
```

Študent opravuje príkaz na:

```
1 == slova[0]
```

ale slovne komentuje, že si je vedomý nesprávnosti riešenia. Chápe a slovne vysvetľuje rozdiel medzi priraďovacím príkazom = a relačnou operáciou ==, ale kvôli nedostatku iných nápadov skúša riešenie, ktoré je navrhované v chybovej správe. V tomto prípade teda nejde o miskoncepciu, ale

chybu vyplývajúcu z chýbajúcich vedomostí o syntaxi premenných v jazyku Python. Učiteľ ich doplní.

Uvedomenie si, že interpreter musí vedieť rozpoznať, čo je hodnota a čo je meno premennej, vedie študentov k hlbšiemu porozumeniu syntaxe jazyka, prečo sa reťazce píšu do apostrofov alebo úvodzoviek a prečo premenné nemôžu mať tvar čísla.

4.3 Návratová hodnota funkcie

V študentskom programe je definovaná funkcia na kreslenie vlajky so škandinávskym krížom:

Program:

```
import tkinter
platno = tkinter.Canvas(width = 500, height = 400)
platno.pack()
def vlajka(farba1, farba2, farba3):
    platno.create_rectangle(100,100,400,300,fill = farba1, outline = '')
    platno.create_rectangle(180,100,220,300,fill = farba2, outline = '')
    platno.create_rectangle(100,180,400,220,fill = farba2, outline = '')
    platno.create_rectangle(190,100,210,300,fill = farba3, outline = '')
    platno.create_rectangle(100,190,400,210,fill = farba3, outline = '')
```

Volaním funkcie `vlajka` s rôznymi hodnotami parametrov `farba1`, `farba2`, `farba3` sa dajú nakresliť vlajky viacerých škandinávskych krajín. Volania funkcie s rôznymi parametrami farieb demonštrovala študentka takto:

```
norsko = vlajka("crimson", "white", "navy")
dansko = vlajka("red", "white", "white")
```

Funkcia `vlajka` je bez návratovej hodnoty, premenným `norsko`, `dansko` sa priradí hodnota `None`. U študentky sa odhalila miskoncepcia, že grafický výstup (obrázok) považuje za výsledok volania funkcie, ktorý je možné priradiť do premennej.

Rovnaká miskoncepcia sa objavila aj v súvislosti s funkciou `print` bez návratovej hodnoty. Študent mal zistiť, akú hodnotu má premenná `vstup[0]` po vykonaní príkazu:

```
vstup = input('Zadaj meno a priezvisko: ').split(' ')
```

Zistil to kontrolným výpisom

```
print(vstup[0])
```

Zistenú hodnotu chcel priradiť premennej s názvom `meno`:

```
meno = print(vstup[0])
```

Za výsledok volania funkcie považoval efekt funkcie `print` – výpis do štandardného výstupu. Príklady ukazujú, že slová výsledok a návratová hodnota funkcie nemajú celkom rovnaký význam a ich nesprávne použitie môže viesť k miskoncepcii.

4.4 Vetvenie

Študent naprogramoval riešenie úlohy o výpočte indexu telesnej hmotnosti nasledovne:

```
vaha = int(input(""))
vyska = float(input(""))
bmi = (vaha / vyska**2)
if bmi < 18.5:
    print("podvýživa")
elif 18.5 < bmi < 25.0:
    print("normálna hmotnosť")
```

```
elif 25.0 < bmi < 30.0:
    print("mierna nadváha")
elif 30.0 < bmi < 40.0:
    print("obezita")
else:
    print("ťažká obezita")
```

Na otázku učiteľa *„Aký bude výstup, keď bude výsledok výpočtu práve 18.5?“* študent nevedel hneď odpovedať. Vykonal experiment a s prekvapením zistil, že program vypísal reťazec `"ťažká obezita"`. V programe následne opravil všetky operátory < na <=.

Študenta sme vyzvali ešte raz prečítať zápis časti programu s vetvením. Študent začal komentovať svoj zdrojový kód nasledovne: *„Ak je `bmi` do 18.5, vypíšeme prvú vec, inak skontrolujeme, či nejde o hodnotu z druhého intervalu a vypíšeme druhú vec...“*.

Na otázku učiteľa *„Akú hodnotu má premenná `bmi`, keď táto hodnota nie je menšia ako 18.5?”* študent odpovedal správne: *„Viac alebo práve toľko.“* Na doplňujúcu otázku *„Nie sú teda niektoré časti podmienok v zápise vetvenia prebytočné?“* študent reagoval zamietavo s vysvetlením, že *„veď program musí overiť všetky možnosti“*.

Z vyjadrenia študenta možno usúdiť, že v zásade rozumie, ako prebieha tok výpočtu v príkaze `if-elif-else`, vedel negovať jednoduchú podmienku a opraviť program pre hraničné hodnoty premennej `bmi`. Pri otázke o možnom zjednodušení zložených podmienok sa však ukázalo, že jeho predstava je naivná a príkaz chápe zjednodušene ako viacnásobné vetvenie s alternatívnymi podmienkami, ktoré sú navzájom nezávislé, a nie ako vnorené vetvenie.

4.5 Cyklus

Študentský program vypisuje na obrazovku prázdny štvorec – okienko – s obrysom so znakov `'x'` s p riadkami a p znakmi v každom riadku:

```
p = int(input("Zadaj veľkosť okienka: "))
for i in range(1,p):
    if i == 1:
       print("x" * p)
    else:
       print("x" + (" " * (p-2)) + "x")
    if i == p-1:
       print("x" * p)
```

Riešenie je správne, ale veľmi neštandardné: `p` riadkov sa vypisuje v `p-1` iteráciách cyklu `for` (v poslednej iterácii dva riadky). Riešenie má v sebe náznak deklaratívneho prístupu k riešeniu úlohy – v tele cyklu sa definuje, čo sa má v akých prípadoch vypísať, ale použitá je postupnosť dvoch podmienených príkazov, čo tento koncept narúša.

Učiteľ sa snažil naviesť študenta na iné systémovejšie a efektívnejšie riešenie rozložením problému na časti, ktoré sa opakujú, a ktoré sa neopakujú: *„V tele cyklu `for` nepotrebujeme použiť príkaz vetvenia. Riadky s medzerami a znakom ‘x’ na začiatku a na konci sa opakujú, preto je vhodné použiť príkaz `for`. Prvý a posledný riadok výstupu však budeme potrebovať vypísať len raz. Kedy vypíšeme prvý riadok a kedy posledný riadok?“*

Študent nevedel zapísať riešenie inak, opakovane sa vracal k argumentu, že riešenie je predsa správne, keď funguje. Prípad je ukážkou vytvorenia a lipnutia na alternatívnej predstave riešenia problému, ktorá je nesystémová. Študent si síce uvedomuje, že prvý a posledný riadok má riešiť ako

osobitné prípady, ale štruktúra jeho programu je komplikovanejšia. Je možné, že správne riešenie úlohy bolo výsledkom nesystematického upravovania programu, kým nedával správne výsledky.

Učiteľovi sa nepodarilo presvedčiť študenta, aby rozmýšľal aj nad iným riešením, lebo metodický postup, ktorý použil, nenechal študentovi dostatok priestoru na vlastný návrh riešenia.

5 DISKUSIA A ZÁVER

Uvedené ukážky didaktických situácií obsahujú príklady rôznych typov chýb v konceptuálnom porozumení v rôznych typoch vedomostí z programovania. Quin a Lehman [4] ich v prehľadovom článku kategorizujú na vedomosti o syntaxi (*syntactic knowledge*), konceptuálne vedomosti o algoritmizácii (*conceptual knowledge*) a vedomosti o stratégiách algoritmického riešenia problémov (*strategic knowledge*). Prípady opísané v 4.1, 4.2, 4.3 obsahujú príklady miskoncepcií týkajúcich sa vedomostí o syntaxi. V prípade 4.4 je opísaný príklad naivnej predstavy (prekoncepcia) z oblasti konceptuálnych vedomostí. Prípad 4.5 je ukážkou alternatívnej koncepcie z oblasti vedomostí o stratégiách riešenia problémov.

Naivné predstavy a alternatívne koncepcie nemusia viesť nevyhnutne k chybe v riešení programátorského problému a k nesprávnemu riešeniu, ale predstavujú problém pre budúci rozvoj programátorských vedomostí a schopností žiaka. Miskoncepcie väčšinou vedú k nesprávnemu riešeniu, ktoré sa prejaví aj navonok, ale odstránenie chyby v riešení konkrétnej úlohy málokedy vedie ku korekcii mylnej predstavy, ktorá je väčšinou všeobecnejšia a vyznačuje sa preto veľkou stabilitou. Preto je dôležité, aby učiteľ rozpoznal, či ide o bežnú chybu alebo chybu v konceptuálnom porozumení.

Jednou z príčin vzniku miskoncepcií u žiakov je prenos miskoncepcie z učiteľa na žiaka. Napokon aj príklady miskoncepcií v tomto príspevku sú zozbierané z vyučovania budúcich učiteľov informatiky. Príčinou bývajú aj chyby v metodike vyučovania, napríklad neštandardné pomenovania premenných alebo prezentovanie riešení, ktoré nie sú vzorové. Vzorové riešenia pomáhajú žiakom rozpoznať v riešeniach úloh z programovania vzory a budovať znalosti na vyššej úrovni abstrakcie prepojené do systému. Preto náš budúci záujem v oblasti skúmania problematiky miskoncepcií v programovaní plánujeme zamerať na prácu s učiteľmi informatiky v rámci projektu DiTEdu spoločne s kolegami z UPJŠ v Košiciach.

6 BIBLIOGRAFICKÉ ODKAZY

[1] RUISEL, Imrich. *Múdrosť v zrkadle vekov*. Bratislava: Ikar, 2005. ISBN 80-551-1059-X

[2] HEJNÝ, Milan. *Teória vyučovania matematiky 2*. Bratislava: SPN, 1989. ISBN 80-08-00014-7

[3] MARX Erik, Clemens WITT and Thiemo LEONHARDT. Identifying Secondary School Students' Misconceptions about Machine Learning: An Interview Study. In: *Proceedings of the 19th WiPSCE Conference on Primary and Secondary Computing Education Research (WiPSCE '24)*. New York: Association for Computing Machinery, 2024. Article 6, 10 p. DOI:10.1145/3677619.3678114

[4] QIAN, Yizhou and James LEHMAN. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Trans. Comput. Educ.* 2018,18(1), Article 1, 24 p. DOI: 10.1145/3077618

[5] HAVERLÍKOVÁ, Viera. *Alternatívne predstavy žiakov vo fyzikálnom poznávaní*. Bratislava: Knižničné a edičné centrum FMFI UK, 2013. ISBN 978-80-8147-005-9

[6] ŠTRAUCH, Peter a Jozef HANČ. Kvantitatívna diagnostika miskoncepcií v prírodovednom vzdelávaní. *Edukácia – vedecko-odborný časopis.* 2017, roč. 2, č. 1. s. 291-302. ISSN 1339-8725

[7] GAVALA, Tadeáš a Stanislav LUKÁČ. Diagnostika žiackych miskoncepcií v pravdepodobnosti. *Matematika–Fyzika–Informatika*. 2018, roč. 27, č. 3, s. 180–191. ISSN 1805-7705

[8] CHIODINI, Luca, et al. A Curated Inventory of Programming Language Misconceptions. In: *Proceedings of the 2021 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '21), June 26-July 1, 2021*. New York: Association for Computing Machinery, 2021, p. 380-386. DOI: 10.1145/3430665.3456343

[9] CRUZ IZU, Cheryl, Cheryl POPE and Amali WEERASINGHE. On the Ability to Reason About Program Behaviour: A Think-Aloud Study. In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17).* New York: Association for Computing Machinery, 2017, p. 305–310. DOI: 10.1145/3059009.3059036.

[10] KEPPENS, Jeroen a David HAY. Concept map assessment for teaching computer programming. *Computer Science Education*. 2008, 18(1), p. 31–42. DOI: 10.1080/08993400701864880.

[11] KACZMARCZYK, Lisa C., et al. Identifying student misconceptions of programming. In: *Proceedings of the 41st ACM technical symposium on Computer science education*. New York: Association for Computing Machinery, 2010, p. 107–111. DOI: 10.1145/1734263.1734299

[12] PAIVA, José Carlos, José Paulo LEAL and Álvaro FIGUEIRA. Automated Assessment in Computer Science Education: A State-of-the-Art Review. ACM Transactions on Computing Education. 2022, 22(3), Article 34, 40 p. DOI: 10.1145/3513140.

Žákovská řešení vybraných matematických úloh MO pomocí programování: výsledky výzkumné sondy

Pupil's solutions to selected MO mathematical taksks using programming: results of a research probe

Ladislav Perk[1] Kristýna Vacková[2]
[1]Přírodovědecká fakulta UJEP, Pasteurova 15, 400 96 Ústí nad Labem, Česko
[2]Střední průmyslová a Vyšší odborná škola, Liberec, Masarykova 3, 460 01 Liberec, Česko
ladislav.perk@gmail.com, kristyna.vackova@pslib.cz

EXTENDED ABSTRACT

This article discusses the issue of applying pupil's programming in solving selected types of mathematical tasks from the Mathematical Olympiad of the Czech Republic (MO). We conducted a research probe to map the strategies for solving mathematical tasks using programming. The aim of the research probe was to determine the success rate of solving individual tasks, the difficulty level of solving tasks, and the level of inspiration in model-solved analogous tasks. In the research probe design, we asked ourselves three research questions. The first question was „What is the perceived ease of understanding the task assignment, the perceived ease of creating functional programs, and the level of inspiration for solving the tasks from the author's solution of the tasks in a professional article?". The second question was „What is the level of success in solving individual tasks?". The third question was „What algorithmic constructions do respondents apply when solving the tasks?".

In answering the first research question, the level of understanding varies between tasks. The perceived level of understanding of the tasks reaches the lowest value of 50% and the highest value of 88%. Lower percentage values were found for tasks with more mathematical operations required. The perceived degree of ease of the tasks reaches the lowest value of 50% and the highest value of 70%. The lower values of perceived ease are due to the need to apply more demanding algorithmic constructs to the source code of functional programs in Python. The perceived degree of inspiration of the task solving by the sample tasks is relatively small, reaching percentage values ranging from 18% to 28%. Thus, the respondents were rather uninspired by patterned task solving.

In answering the second research question, it can be noted that the success rate of solving the tasks ranges from 38% to 72% (only programs listing all correct results). It has been shown that the success rate of solving tasks is related to the perceived difficulty of the tasks. The lower success rate of task solving was in tasks with more difficult solutions (e.g., expressing pairs and triples of digits from a five-digit number, finding the largest number from a set of numbers, having to use absolute value to formulate the difference of two numbers, etc.).

In answering the third research question, it can be noted that the respondents applied very diverse coping strategies. For example, they did not use systematic searches to search the state space, but sought solutions by generating random numerical values of control variables. Or they used an informatic approach instead of a mathematical one: they worked with numbers as strings and digits of numbers as characters. More such original approaches to task solving can be found in our research probe.

Keywords

Research probe, programming, source code, program, Python, Math olympiad, math task.

ABSTRAKT

V tomto příspěvku se zabýváme problematikou uplatnění žákovského programování při řešení vybraných typů matematických úloh Matematické olympiády České republiky (MO). Zrealizovali jsme výzkumnou sondu, abychom zmapovali strategie řešení matematických úloh pomocí programování. Cílem výzkumné sondy bylo také zmapování míry úspěšnosti vyřešení jednotlivých úloh, míry snadnosti vyřešení úloh a míry inspirace ve vzorově řešených úlohách.

Výzkumná sonda byla realizována v od začátku května do poloviny června roku 2024. Respondenty výzkumné sondy tvořilo 50 žáků druhých a třetích ročníků oboru Informační technologie na Střední průmyslové škole v Liberci. V dotazníku vlastní konstrukce bylo zadáno celkem pět matematických úloh, kdy zadání úloh bylo inspirováno vybranými matematickými úlohami Matematické olympiády ČR. V rámci zadání úloh byl také uveden odkaz na podobně řešené úlohy v odborném článku prvního z autorů tohoto článku.

Dále byly formulovány tři výzkumné otázky. První otázka byla „Jaká je vnímaná snadnost pochopení zadání úloh, vnímaná snadnost vytvoření funkčních programů a míra inspirace řešení úloh z autorského řešení úloh v odborném článku?“. Druhá otázka byla „Jaká je míra úspěšnosti řešení jednotlivých úloh?“. Třetí otázka byla „Jaké algoritmické konstrukce respondenti při řešení úloh uplatňují?“.

Naše výsledky umožňují nahlédnout, jak respondenti vnímali obtížnost zadání úloh, jak vnímali snadnost vyřešení těchto úloh pomocí programování a zda se inspirovali vzorově řešenými úlohami. Bylo prokázáno, že s rostoucím počtem požadovaných matematických operací v zadáních úloh se zvyšuje vnímaná obtížnost zadání úloh. Dále bylo prokázáno, že se zvyšující se náročností algoritmických konstrukcí (potřebných k vyřešení úloh) se zvyšuje vnímaná obtížnost vyřešení těchto úloh. Zároveň bylo prokázáno, že míra inspirace vzorově řešenými úlohami nebyla výrazně vysoká. Uplatněné strategie řešení úloh respondenty byly velmi různorodé, například v neuplatnění systematického prohledávání stavového prostoru, ale prohledávání stavového prostoru náhodně vybíranými variantami řešení; práce s čísly (místo práce dekadickými zápisy) jako s řetězci a se znaky apod.

Výsledky našich zjištění mohou posloužit učitelům informatiky a matematiky, kteří by rádi využili programování jako nástroj řešení u svých žáků (a to nejen úloh MO), a to zejména pro získání konkrétní představy nejen k otázkám týkajících se úspěšnosti řešení úloh, ale také získání konkrétních představ o uplatněných strategií řešení v podobě algoritmických konstrukcí.

Klíčová slova

Výzkumná sonda, programování, zdrojový kód, program, Python, Matematická olympiáda, matematická úloha.

1 ÚVOD

V současnosti jsme svědky realizace nových pojetí výuk informatik, a to jak na základních, tak středních školách ČR. Od školního roku 2023/2024 se nově zavedená informatika vyučuje na prvním stupni ZŠ, od školního roku 2024/2025 pak na druhém stupni ZŠ. Od školního roku 2025/2026 budou gymnázia vyučovat nově zavedenou informatiku. Nově zavedená informatika se podle RVP člení na následující čtyři okruhy: (1) Data, informace a modelování; (2) Algoritmizace a programování; (3) Informační systémy a (4) Digitální technologie [1, 2].

V našem článku se tematicky zaměříme na druhý z vyjmenovaných okruhů, a to Algoritmizaci a programování. Přestože je obsah tohoto okruhu poměrně v současnosti již dobře zpracován a publikován na internetu, je vhodné se také zabývat dalšími možnostmi uplatnění algoritmizace a programování ve školním prostředí. Jednou z možností je využití programování jako žákovského prostředku řešení matematických úloh.

V pedagogické realitě se lze totiž setkat se žáky, kteří rádi řeší matematické úlohy české a slovenské Matematické olympiády [3, 4]. Tito žáci se ale během řešení těchto úloh mohou dostat do různorodých řešitelských situací: potřebují si ověřit správnost resp. úplnost svých výsledků získaných matematickými prostředky či nedoberou se ke správným výsledkům pomocí matematických prostředků a potřebují tyto výsledky získat například pro abstrakci matematických vzorů. A pokud jsou tyto úlohy řešitelné pomocí systematického experimentování, pak je žádoucí žákům ukázat možnosti řešení těchto úloh pomocí programování.

První z autorů tohoto článku již publikoval odborný článek [5], ve kterém čtenářům přestavil vzorově řešené matematické úlohy matematické olympiády MO pomocí programování v jazyce Python. Součástí článku bylo zadání úloh k samostatnému zpracování, které se následně staly předmětem empirického zkoumání. Z důvodu zájmu o získání empirických poznatků byla zrealizována výzkumná sonda, jejíž výsledky v tomto článku prezentujeme.

Cílem tohoto příspěvku je na základě analýzy zrealizované výzkumné sondy:

- prezentace uplatňovaných algoritmických konstrukcí respondenty výzkumné sondy;
- zhodnotit míru úspěšnosti vyřešení jednotlivých matematických úloh;
- zmapovat subjektivně vyjádřené postoje respondentů k:
    - pochopení zadání úloh;
    - snadnosti vyřešení úloh;
    - inspiraci ve vzorově analogicky řešených úlohách v článku [5].

2 UPLATNĚNÍ PROGRAMOVÁNÍ JAKO PROSTŘEDKU PRO ŽÁKOVSKÉ ŘEŠENÍ VYBRANÝCH MATEMATICKÝCH ÚLOH MO

V této kapitole budou v první části uvedeny argumenty pro využití programování při žákovském řešení matematických úloh MO. Ve druhé části kapitoly bude rozebráno systematické experimentování jako heuristická metoda řešení matematických úloh. Stejně tak bude zmíněna metoda hrubé síly, která je aplikací systematického experimentování při řešení matematických úloh pomocí počítače.

2.1 Argumenty pro a proti využití programování při žákovském řešení vybraných matematických úloh MO

Žáci mohou poměrně úspěšně uplatnit programování při hledání řešení matematických úloh, které jsou řešitelné pomocí systematického experimentování. V případě využití programování jako prostředku žákovského řešení matematických úloh lze uvažovat, že žáci:

- získají prostředek pro řešení vybraných matematických úloh MO;
- posilují kompetenci k řešení problému včetně východiska k případné abstrakci;
- posilují mezipředmětové vztahy mezi informatikou a matematikou;
- mohou zažít úspěch díky úspěšnému nalezení řešení úloh pomocí programování.

Naopak, v pedagogické realitě se lze setkat také se situacemi, kdy žáci:

- nemusí spatřovat dostatečný smysl využití programování při řešení matematických úloh;
- nemají dostatečně rozvinutou dovednost programovat;
- nemají chuť učit se novým věcem;
- mohou pociťovat strach z neúspěchu.

2.2 Systematické experimentování jako metoda řešení matematických úloh

Žáci mohou velmi dobře uplatnit programování při hledání řešení těch matematických úloh, které jsou řešitelné pomocí systematického experimentování. V takových úlohách se převážně pracuje s přirozenými, resp. celými čísly. Protože předmětem zkoumání naší výzkumné sondy jsou

matematické úlohy, které byly řešeny pomocí systematického experimentování, v krátkosti se v následujících řádcích touto metodou budeme zabývat.

Systematické experimentování představuje jednu z experimentálních metod při řešení matematických úloh. Tato metoda spočívá v tom, že požadovaného výsledku lze dosáhnout prostřednictvím organizovaného a postupného provádění pokusů, kdy každý následující pokus je upraven o určitý krok. Základní princip vychází z postupného přibližování se k cílovému řešení od výchozí hodnoty [7, 8, 9, 10]. Pokud je při realizaci systematického experimentování využit počítač, hovoříme o tzv. metodě hrubé síly [8, 9]. Tento přístup lze definovat jako systematické zkoumání všech možných variant v rámci množiny potenciálních výsledků [7, 8, 10]. Pro účely tohoto článku budeme označovat proces postupného a systematického zkoumání jednotlivých možností v rámci množiny všech potenciálních výsledků jako prohledávání stavového prostoru či průchod stavovým prostorem.

3 METODIKA

Tato kapitola popisuje metodiku realizované výzkumné sondy. V první části jsou formulovány cíle výzkumné cíle sondy a následně z nich formulované výzkumné otázky. Ve druhé části je uvedena realizace výzkumné sondy. Ve třetí části je v krátkosti popsán dotazník vlastní konstrukce, který byl respondentům v rámci výzkumné sondy předložen. Ve čtvrté - největší části kapitoly - jsou uvedeny zadání všech pěti úloh, příslušné zdrojové kódy v jazyce Python a výpisy programů těchto zdrojových kódů.

3.1 Cíle výzkumné sondy, výzkumné otázky

Cílem výzkumné sondy je zmapovat řešitelské strategie respondentů výzkumné sondy při řešení pěti předložených úloh; zmapovat míru vnímané snadnost řešení úloh; zmapovat míru úspěšnosti řešení těchto úloh a dále zmapovat míru inspirace řešení respondentů řešeními uvedenými v odborném článku [5].

Na základě výzkumných cílů jsme zformulovali následující výzkumné deskriptivní otázky [6]:

Výzkumná otázka č. 1: Jaká je vnímaná snadnost pochopení zadání úloh, vnímaná snadnost vytvoření funkčních programů a míra inspirace řešení úloh z autorského řešení úloh v odborném článku?

Výzkumná otázka č. 2: Jaká je míra úspěšnosti řešení jednotlivých úloh?

Výzkumná otázka č. 3: Jaké algoritmické konstrukce (řešitelské strategie) respondenti při řešení úloh uplatňují?

3.2 Realizace výzkumné sondy

Respondenty výzkumné sondy tvořili žáci druhých a třetích ročníků Střední průmyslové školy a Vyšší odborné školy Liberec, studijního oboru Informační technologie. Tito respondenti již absolvovali alespoň jeden školní rok výuky programování, konkrétně v jazyce Python.

Nejprve se uvažovalo o realizaci dotazníkového šetření s využitím online aplikace Google Forms, ale nakonec se přistoupilo k realizaci výzkumné sondy pomocí vytištěných dotazníků v papírové podobě.

Celkem bylo distribuováno 54 dotazníků, navráceno bylo všech 54 dotazníků, návratnost dotazníku byla tedy 100 %. Řádně bylo vyplněno 50 dotazníků, 4 dotazníky byly vyřazeny pro neúplnost jejich vyplnění. Výsledky výzkumné sondy se opírají o zjištění z těchto 50 řádně vyplněných dotazníků. Respondentům byly vytištěné dotazníky předány na začátku května 2024 a termín pro jejich odevzdání byl v polovině června 2024.

3.3 Dotazník vlastní konstrukce

Dotazník vlastní konstrukce obsahoval celkem pět úloh. Každá z těchto úloh obsahovala pole pro vypsání funkčního zdrojového kódu programu v jazyce Python, dále pak dvě doplňkové otázky, které se týkaly vyjádření míry pochopení zadání úlohy a vnímané snadnosti napsání funkčního programu. Třetí doplňkovou otázkou byla otázka týkající se vyjádření míry inspirace respondentského řešení úlohy s autorským řešením analogicky podobné úlohy.

3.4 Představení úloh výzkumné sondy

V této části představíme pět testových úloh, které byly předloženy respondentům výzkumného šetření k řešení. Součástí zadání úloh je také funkční zdrojový kód v jazyce Python, který je prezentován jako autorské řešení. Po uvedených zdrojových kódech jsou vypsány výstupy, které programy těchto zdrojových kódů vypisují.

■ Úloha 1. Eva má čtyři papírky a na každém z nich je napsáno jedno přirozené číslo. Když vynásobí mezi sebou všechny trojice čísel z papírků, dostane výsledky 30, 36, 60 a 90. Která čísla jsou napsána na Eviných papírcích? ([5], str. 34)

Autorské řešení.

```
for a in range(1, 91):
    for b in range(1, 91):
        for c in range(1, 91):
            for d in range(1, 91):
                abc = a * b * c
                abd = a * b * d
                acd = a * c * d
                bcd = b * c * d

                if abc == 30 and abd == 36 and acd == 60 and bcd == 90:
                    print("Nalezena ctverice: ", a, ",", b, ",", c, ",", d)
                    print()

print("Konec")
```

Obrázek 1: Zdrojový kód autorského řešení úlohy 1

Program vypíše jednu čtveřici čísel, a to 2, 3, 5 a 6. Výpočtem lze snadno ověřit, že uvedená čísla jsou správná.

□

■ Úloha 2. Součin čtyř přirozených čísel je 120. Kdybychom první z nich zvětšili o 5, zvětšil by se součin o 200. Kdybychom druhé z nich zvětšili o 5, zvětšil by se o 150. Kdybychom třetí z nich zmenšili o 1, zmenšil by se o 60. Kdybychom čtvrté z nich zvětšili o 10, zvětšil by se na trojnásobek původní hodnoty. Která čtyři přirozená čísla mají tuto vlastnost? ([5], str. 37)

Autorské řešení.

```
for a in range(1, 121):
    for b in range(1, 121):
        for c in range(1, 121):
            for d in range(1, 121):
                soucin = 120

                if a * b * c * d == soucin and (a + 5)* b * c * d == soucin + 200 and
                a * (b+5) * c * d == soucin + 150 and a * b * (c-1) * d == soucin - 60 and
                a * b * c *(d+10) == 3 * soucin:
                    print("Nalezena ctverice: ", a, ",", b, ",", c, ",", d)
```

Obrázek 2: Zdrojový kód autorského řešení úlohy 2

Program vypíše jednu čtveřici čísel (a, b, c, d): (3, 4, 2 a 5).

□

■ Úloha 3. Uvažme pětimístné přirozené číslo s následující vlastností: jestliže prohodíme jeho první trojčíslí s posledním dvojčíslím, dostaneme pětimístné číslo o 14 769 menší. Kolik je takových čísel celkem? ([5], str. 39)

Autorské řešení.

```
pocet = 0

for abc in range(100, 1000):
    for de in range(10, 100):
        abcde = abc * 100 + de
        deabc = de * 1000 + abc

        if abcde == deabc + 14769:
            print(abcde)
            pocet = pocet + 1

print("Nalezeny pocet: ", pocet)
```

Obrázek 3: Zdrojový kód autorského řešení úlohy 3

Program vypíše jako řešení celkem šest hledaných pěticiferných čísel: 35 120, 46 231, 57 342, 68 453, 79 564 a 90 675. Každé z těchto šesti čísel odpovídá podmínkám zadání úlohy.

□

■ Úloha 4. Monika přemýšlí o největším pětimístném čísle, které má následující vlastnosti: součin prvních dvou a posledních dvou cifer je 448; cifra uprostřed je menší než 6; rozdíl cifer první dvojice cifer je dvakrát větší než rozdíl cifer poslední dvojice cifer; rozdíl myšleného čísla a opačně napsaného čísla je největší možný. Určete Moničino myšlené číslo. ([5], str. 42)

Autorské řešení.

```
max = 0

for a in range(1, 10):
    for b in range(0, 10):
        for c in range(0, 10):
            for d in range(0, 10):
                for e in range(1, 10):
                    abcde = a * 10000 + b * 1000 + c * 100 + d * 10 + e
                    edcba = e * 10000 + d * 1000 + c * 100 + b *10 + a
                    rozdil = abcde - edcba

                    if a * b * c * d * e == 448 and c < 6 and abs(a - b)==2*(abs(d - e)):
                        if rozdil > max:
                            max = rozdil
                            cislo = abcde

print("Nalezene cislo: ", cislo)
print("Max rozdil: ", max)
```

Obrázek 4: Zdrojový kód autorského řešení úlohy 4

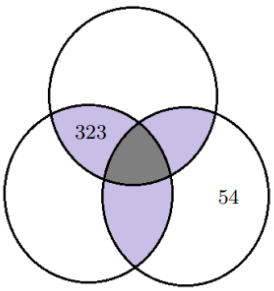
Program vypíše jako řešení nalezené Moničino pěticiferné číslo, a to 82 574. Stejně tak vypíše rozdíl Moničina čísla a opačně zapsaného čísla, kterým je číslo 35 046.

Poznámka. V případě, že by respondenti místo neostré nerovnosti v zápisu **if rozdil >= max:** (ř. 13) uplatnili ostrou nerovnost **if rozdil > max:**, pak by program vypsal jako největší nalezené číslo 82 074. Pěticiferná čísla 82 074, 82 174, 82 274, 82 374, 82 474 a 82 574 totiž mají stejný hledaný

číselný rozdíl s jejich opačně zapsaným číslem, a to číslo 35 046. Při uplatnění ostré nerovnosti by se uplatnilo pouze číslo 82 074, další výše zmíněná čísla by již vzhledem ke shodnosti číselného rozdílu již jako největší hledaná pěticiferná čísla nemohla registrovat.

□

■ Úloha 5. Do prázdných polí v následujícím obrázku doplňte celá čísla větší než 1 tak, aby v každém tmavším políčku byl součin čísel ze sousedních světlejších políček: Jaké je číslo ve středu? ([5], str. 44)



Obrázek 5: Grafické vyjádření číselných vztahů v zadání úlohy 5 ([5], str. 44)

Autorské řešení.

```
for a in range(2, 324):
    for b in range(2, 324):
        for c in range(54, 55):
            ab = a * b
            bc = b * c
            ac = a * c
            abc = ab * bc * ac

            if ab == 323:
                print("abc = ", abc)
                print("a = ", a)
                print("b = ", b)
                print("c = ", c)
                print()
```

Obrázek 6: Zdrojový kód autorského řešení úlohy 6

Program vypíše jako řešení třináctkrát číslo 304 223 364. Zároveň informativně vypíše výsledek 304 223 364 a zároveň dvě trojice čísel (17, 19, 54), (19, 17, 54).

□

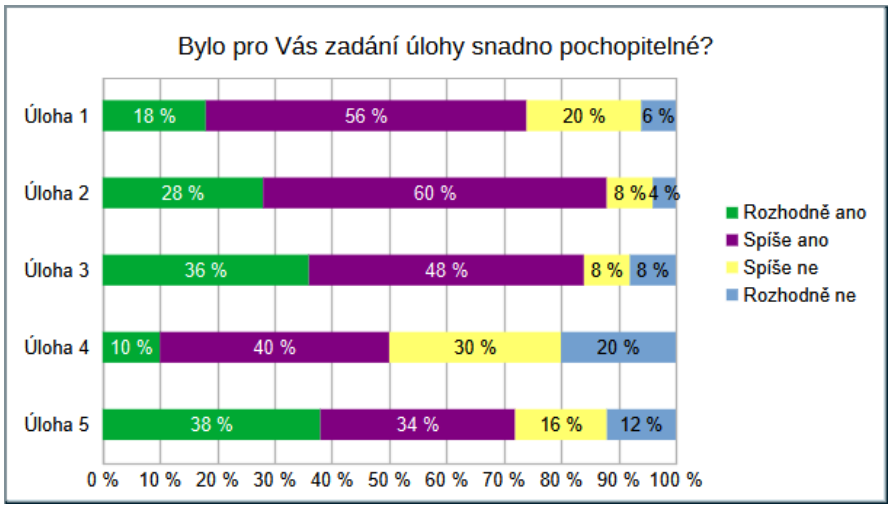
4 VÝSLEDKY VÝZKUMNÉ SONDY

Následující kapitola je stěžejní kapitolou celého článku. Uvádí a komentuje výsledky realizované výzkumné sondy. V první části odpovídá na výzkumnou otázku č. 1, kdy zhodnocuje míru vnímané snadnosti pochopení zadání úloh, míru vnímané snadnosti řešení úloh pomocí programování a míru inspirace respondentů řešeními uvedenými v odborném článku [5]. Ve druhé části pak odpovídá na výzkumnou otázku č. 2, kdy stanovuje míru úspěšnosti vyřešení jednotlivých úloh. Ve třetí – nejobsáhlejší části kapitoly – pak uvádí a rozebírá algoritmické konstrukce, které respondenti v rámci svých řešení uplatnili.

4.1 Zhodnocení výzkumné otázky č. 1

Zhodnocení vnímané snadnosti pochopení zadání úloh respondenty

V první doplňkové otázce byli respondenti dotazováni na vnímanou snadnost pochopení zadání úloh. Pro tento účel byla respondentům předložena otázka „Bylo pro Vás zadání úlohy snadno pochopitelné?“ se čtyřstupňovou škálou. Z grafu (viz obrázek 7) je patrné, že nejvíce pochopitelné byly pro respondenty úlohy 2 a 3, které vykazovaly míru pochopení zadání 88% a 84% (součet odpovědí „Rozhodně ano“ a „Spíše ano“).



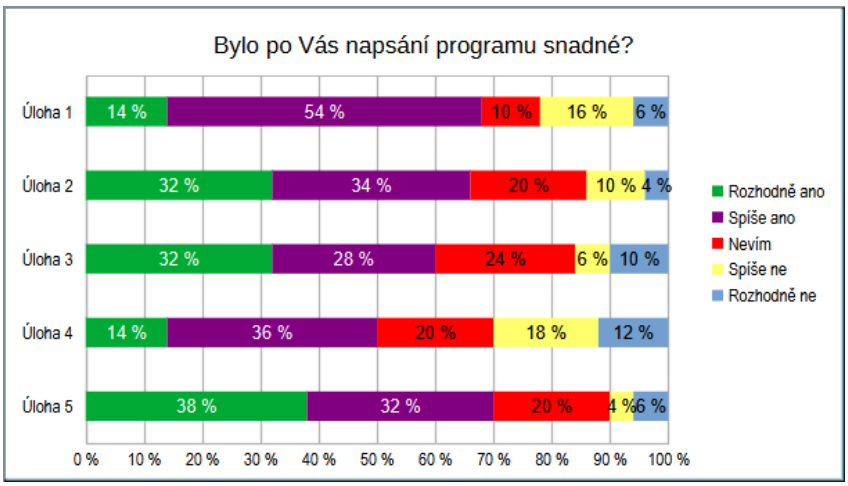
Obrázek 7: Vyjádření vnímané snadnosti pochopení zadání jednotlivých úloh respondenty

V úloze 2 pracuje se čtyřmi přirozenými čísly a v zadání se explicitně formulují konkrétní početní úkony vztahující se na každou z nich. Analogicky stejně se v úloze 3 formulují početní úkony s prvním trojčíslím a druhým dvojčíslím pěticiferného čísla, jedná se o zcela explicitně vyjádřené početní úkony. Je pravděpodobné, že díky této explicitnosti vyjadřovaných úkonů proto úlohy 2 a 3 dosahují vysoké míry pochopení zadání.

Velmi podobnému procentuálnímu vyjádření míry pochopení zadání dosahují úlohy 1 a 5 (74% a 72%). V případě úlohy 1 může být pro respondenty obtížné vyjádřit všechny možné číselné trojice ze čtyř čísel bez opakování. Určitým překvapením je vyjádření relativně nižší míry pochopení úlohy 5. Úloha 4 vykazuje nejnižší míru pochopení svého zadání, a to 50%.

Zhodnocení vnímané snadnosti napsání funkčních zdrojových kódů programů respondenty

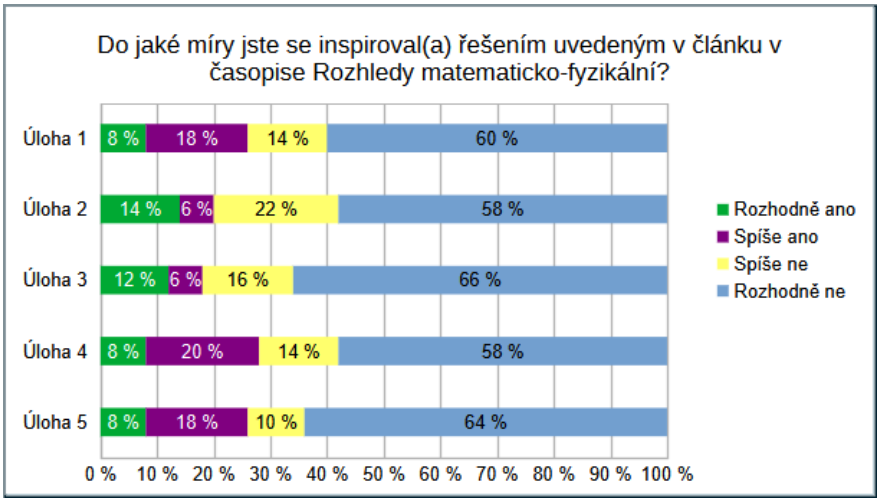
Ve druhé doplňkové otázce byli respondenti dotazováni na vnímanou snadnost napsání funkčního programu. Pro tento účel byla respondentům předložena otázka „Bylo pro Vás napsání programu snadné?“ s pětistupňovou škálou. Z grafu (viz obrázek 8) je patrné, že respondenti vnímali u úlohy 1 (68 %) a překvapivě u úlohy 5 (70 %) nejvyšší snadnost napsání funkčního programu; v těsném závěsu pak u úlohy 2 (66 %). Vyšší obtížnost napsání funkčního programu pak respondenti vnímali u úlohy 3 (60 %). Očekávaně pak nejvyšší obtížnost napsání funkčního programu vnímali respondenti u úlohy 4 (50%).



Obrázek 8: Vyjádření vnímané snadnosti vyřešení úloh napsáním funkčních programů jednotlivých úloh respondenty

Zhodnocení míry inspirace řešení z autorského článku respondenty

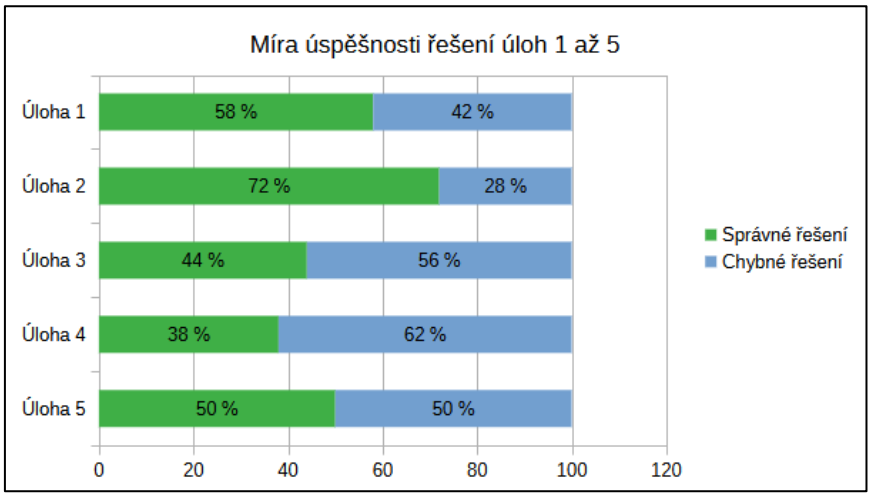
Ve třetí doplňkové otázce byli respondenti dotazováni na míru inspirace svého řešení autorským řešením v [5]. Pro tento účel byla respondentům předložena otázka „Do jaké míry jste se inspiroval(a) řešením uvedeným v článku v časopise Rozhledy matematicko-fyzikální?“ s čtyřstupňovou škálou. Z grafu (viz obrázek 8) je patrné, že míra inspirace řešení odborným článkem [5] je u všech úloh relativně nízká, dosahuje hodnot od 18% do 28%. Lze tedy předpokládat, že přibližně tři čtvrtiny respondentů uvede originální řešení úloh.



Obrázek 9: Vyjádření míry inspirace autorským řešením prezentovaným v odborném článku [5]

4.2 Zhodnocení výzkumné otázky č. 2

Cílem této sekce bylo zjistit míru úspěšnosti řešení předložených úloh. Za správné řešení úlohy jsme považovali takové řešení, které vypisovalo správné výsledky; v případě většího počtu požadovaných výsledků pak všechny výsledky. Všechna ostatní řešení byla považována za chybná. Z grafu (viz obrázek 10) je zřejmé, že respondenti byli z hlediska správnosti napsání funkčních programů nejúspěšnější v úloze 2 (72 %). Druhou v pořadí nejvíce řešenou úlohou respondenty je úloha 1 (58 %). Třetí nejúspěšněji řešená úloha 5 (50%). Druhá nejméně úspěšně řešená úloha je úloha 3 (44 %). Nejméně úspěšně řešenou úlohou je úloha 4 (38 %).



Obrázek 10: Zhodnocení míry úspěšnosti vyřešení úloh výzkumné sondy

4.3 Zhodnocení výzkumné otázky č. 3

Pro úspěšné zodpovězení výzkumné otázky 3 byla provedena podrobná analýza vytvořených zdrojových kódů z odevzdaných dotazníků všech pěti úloh. Zajímaly nás algoritmické konstrukce, které respondenti pro účely svých řešení uplatňovali.

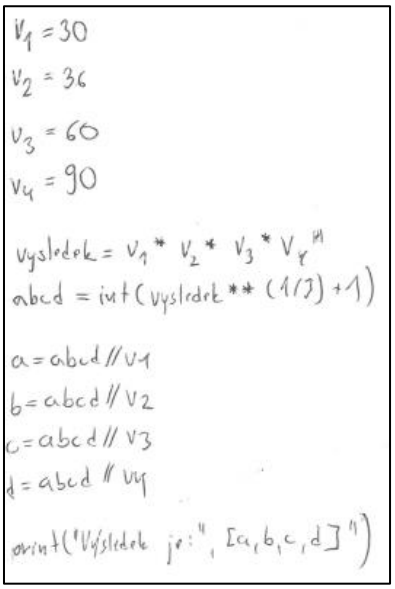
Následně uvidíme, že respondenti uplatnili velmi různorodé strategie řešení. S ohledem na skutečnost, že variabilita dílčích fází řešení je poměrně vysoká, je v rámci prezentace výsledků uvedeno procentuální zastoupení jednotlivých fází strategií řešení.

Úloha 1

V rámci provedené analýzy úlohy 1 byly zjištěny následující poznatky:

- Vytvořený zdrojový kód respondentem je shodný či analogicky podobný zdrojovému kódu v autorském řešení úlohy (32 %).
- Všechny podmínky pro výpis číselných hodnot byly vyjádřeny zvlášť v metodě, která byla volána v hlavním těle programu (10 %).
- Každá z dílčích podmínek je obsažena ve vnořených podmíněných příkazech *if*; po výpisu řešení pomocí *print()* využito *break* (10 %).
- Využití *while* cyklu v zápisu *while True:*, kdy pro účely vyčíslení použita metoda *random.randrage()* s krokem -1, popřípadně metoda *random.randint()* (6 %).
- Zbytečné nulování řídících proměnných, které byly následně použity jako řídící proměnné vzájemně vnořených cyklů (4 %).
- Využití metody *solve()* z knihovny *sympy* (2 %).
- Využití metody *combinations()* z knihovny *itertools*; využití proměnné *a* jako list se čtyřmi prvky 0 až 3; práce s *a[0]* až *a[3]* (2 %).

Zároveň je žádoucí sdělit, že 18 % respondentů vyřešilo úlohu zcela originálním způsobem: přímým výpočtem. Tím se vyhnuli prohledávání stavového prostoru. Za všechna tato řešení představujeme jedno zástupné na obrázku 11.



Obrázek 11: Ukázka respondentského řešení pomocí přímého vzorce

Uvedeme proto vysvětlení uvedeného postupu řešení: Nejprve vynásobili čísla všech číselných trojic *a.b.c* . *a.b.d* . *a.c.d* . *b.c.d*, které se po úpravě rovná a^3 . b^3 . c^3 . d^3. Zároveň je tento součin roven součinu cifer všech číselných trojic, tzn. číslu 30 . 36 . 60 . 90 = 5 832 000. Proto platí a^3 . b^3 . c^3 . d^3 = 5 832 000, po odmocnění třetí odmocninou pak platí *a* . *b* . *c* . *d* = 180. Z prvního čísla *a* . *b* . *c* = 30 musí platit *d* = 6, z druhého čísla *a* . *b* . *d* = 36 musí platit *c* = 5, z třetího čísla *a* . *c* . *d* = 60 musí platit *b* = 3, a z čtvrtého čísla *b* . *c* . *d* = 90 musí platit *a* = 2. Respondenti tak získali tak hledaná čísla 2, 3, 5 a 6.

Úloha 2

V rámci provedené analýzy úlohy 2 byly zjištěny následující poznatky:

- Vytvořený zdrojový kód respondentem je shodný či analogicky podobný zdrojovému kódu v autorském řešení úlohy (48 %).
- Ve zdrojovém kódu byly dílčí podmínky formulovány pomocí vzájemně vnořených podmíněných příkazů *if* zvlášť, bez využití konjunktivních spojek *and* (22 %).
- Celý program byl definován jako metoda, bez uvedení volání metody (12 %).
- Řídící proměnné *a, b, c, d* byly formulovány jako list; po podmínce *a.b.c.d == 120* naplnění probíhalo pomocí *for* cyklů s využitím metody *.append([a, b, c, d])*; dále se pokračovala práce s proměnnými jako indexovanými prvky listu od 0 do 3; posouzení splnění jednotlivých podmínek byly formulovány pomocí systému *if – continue* (4 %).
- Byl využit cyklus *while True:*, kdy čtyři proměnné byly vyčíslovány pomocí *random.randint()* a celý proces trval do prvního nalezeného řešení (4 %).
- Byl využit cyklus *while True:*, kdy čtyři proměnné byly vyčíslovány pomocí *random.randrage()* a celý proces trval do prvního nalezeného řešení (2 %).
- Použití metod *solve()*, *Eq()* a *solve()* z knihovny *sympy* (2 %).

Úloha 3

V rámci provedené analýzy úlohy 3 byly zjištěny následující poznatky:

- Vytvořený zdrojový kód respondentem je shodný či analogicky podobný zdrojovému kódu v autorském řešení úlohy (10 %).
- Práce s čísly jako s řetězci: nejprve přirozené číslo přetypováno na řetězec pomocí metody *str()*, výsledné číslo vzniklo jako spojení dílčích cifer v příslušném pořadí pomocí příslušné

indexace řetězce znamenající pěticiferné číslo; výsledný řetězec přetypován na číslo pomocí metody metody *int()* (34 %).

- Práce s čísly jako s řetězci analogicky podobně jako v předchozím případě: nejprve přirozené číslo přetypováno na řetězec pomocí metody *str()*, extrahování cifer jako řetězců pomocí *[:3]* a *[3:]* (v pěticiferném čísle první tři cifry a poslední dvě cifry); následné spojení obou řetězců do jednoho řetězce a v konečné fázi výsledný řetězec přetypován na číslo pomocí metody *int()* (26 %).
- Proměnné *a* až *e* vyčísleny jako cifry pomocí pěti vzájemně vnořených *for* cyklů. Hledané číslo zformulováno jako *int(str(a)+ str(b)+ str(c)+ str(d)+ str(e))* a pozměněného čísla *int(str(d)+ str(e)+ str(a)+ str(b)+ str(c))* (8 %).
- Originální a nápaditý způsob nalezení řešení pomocí celočíselného dělení: (a) první trojici cifer vyjádřit jako trojciferné číslo pomocí celočíselného dělení číslem 100; (b) poslední dvojici čísel jako rozdíl posuzovaného pěticiferného čísla a stonásobku první trojice čísel; (c) vzniklé číslo vyjádřit jako součet stonásobku poslední dvojice čísel a trojciferné číslo z bodu (a). Konkretizovaná ukázka: pěticiferné číslo 46 231: viz (a) 462; viz (b) 46 231 – 462 . 100 = 31; viz (c) 31 . 100 + 462 = 31 462. Rozdíl 46 231 – 31 462 je požadovaným rozdílem 14 769 (6 %).
- Další zajímavý způsob řešení, avšak náročnější: využití metod *str()*, *.join()* a *map()*: vyčíslení proměnných *i* , *j*, *k* , *l* , *m* jako cifry pomocí pěti vzájemně vnořených *for* cyklů, následné zformulovaní hledaného pěticiferného čísla jako *' '. join(map(str, [i,j,k,l,m]))* a otočeného čísla *' '. join(map(str, [l,m,i,j,k])*; posuzován rozdíl přetypovaného hledaného čísla pomocí *int()* a přetypovaného otočeného čísla pomocí *int()* (4 %).
- Pěticiferné číslo *abcde* je zformulováno (po vyčíslení cifer *a* až *e* pomocí vzájemně vnořených *for* cyklů) pomocí *int(f"{a}{b}{c}{d}{e}")*; pěticiferné číslo *deabc* je pak zformulováno pomocí *int(f"{d}{e}{a}{b}{c}")* (4 %).
- Realizace ciferného rozkladu pomocí celočíselného dělení a operace modulo: Proměnná *cisla* vyčíslena jako pěticiferné číslo pomocí *for* cyklu; cifra reprezentující desetitisíce jako *cisla // 10000*, cifra reprezentující tisíce jako *(cisla // 1000) % 10*, cifra reprezentující stovky jako *(cisla // 100) % 10*, cifra reprezentující desítky jako *(cisla // 10) % 10* a cifra reprezentující jednotky jako *cisla % 10* (2 %).

Úloha 4

V rámci provedené analýzy úlohy 4 byly zjištěny následující poznatky:

- Vytvořený zdrojový kód respondentem je shodný či analogicky podobný zdrojovému kódu v autorském řešení úlohy (12 %).
- Využití předčasného ukončení prohledávání stavového prostoru pomocí systému *if – break* (22 %).
- Vyčíslování řídících proměnných tvořících jednotlivé cifry pěticiferného čísla sestupně: např. *for a in range (9, 0, -1); for b in range (9, 0, -1)* apod. (8 %).
- Každá z pěti proměnných formulována jako *int(str(number) [0])*, … *int(str(number) [4])*, kde proměnná *number* je vyčíslována pomocí *for* cyklu od 10 000 do 99 999 (6 %).
- Prohledávání stavového prostoru pomocí *nums=[random.randrage(1,9), random.randrage(0,9), random.randrage(0,9), random.randrage(0,9), random.randrage(0,9)]*, a následná práce s *nums[0]* až *nums[4]*; využití metod *' '. join(map(str, [nums[4], nums[3], nums[2], nums[1], nums[0]]))* (4 %).
- Práce s jednotlivými ciframi pěticiferného čísla, které vzniknou přetypováním pěticiferného čísla *i* na řetězec *s = str(i)* a pomocí indexace pak práce s těmito ciframi: *s[0]* až *s[4]* (4 %).
- Posouzení splnění jednotlivých podmínek bylo formulováno pomocí systému *if – continue*. (4 %).

- Originální úspěšné řešení úlohy: Prohledávání stavového prostoru a vyčíslení dílčích cifer pěticiferného čísla realizováno: na prvním řádku: *for y in range(99999,9999,-1):*, na druhém řádku vnořeně pak *i = [int(x) for x in str(y)]* (2 %).

Je třeba sdělit, že respondenti měli v nemalé míře obtíže správně algoritmicky vyjádřit největší číselný rozdíl a nalézt nejmenší hodnotu tohoto rozdílu.

Úloha 5

V rámci provedené analýzy úlohy 5 byly zjištěny následující poznatky:

- Vytvořený zdrojový kód respondentem je shodný či analogicky podobný zdrojovému kódu v autorském řešení úlohy (18 %).
- Zúžení velikosti prohledávaného stavového prostoru: Pro účely hledání číselných hodnot proměnných *a* a *b*, jejichž součin je roven 323, byl uplatněn jeden *for* cyklus od 2 do $\sqrt{ab}$, nikoliv do *ab*. Bez újmy správnosti řešení se zúžila velikost stavového prostoru (2 %).
- Pro účely vyčíslení proměnných *a* a *b* byla uplatněna metoda *combinations()* z knihovny *itertools* (2 %).
- Hledání největšího dělitele čísla 323: Pro účely nalezení číselných hodnot proměnných *a* a *b*, jejichž součin je roven 323, byla uplatněna konstrukce *i = 2*; na dalším řádku *while True:*, na dalším řádku vnořeno *if 323 % i==0*, poté *cislo = 323 / i* a *break*; úrovní pod *if* pak následovalo *i=i+1*. Tím autor této konstrukce získal číslo 19. Číslo 17 získal poměrem 323 / 19 (2 %).
- Hledání všech celočíselných dělitelů čísel 323 a 54, jejich postupné dosazování do proměnných *a*, *b*, *c* a následné posuzování splnění všech podmínek kladená na *a*, *b*, *c* (2 %).
- Použití metod *solve()*, *Eq()* a *solve()* z knihovny *sympy* (2 %).

5 DOPORUČENÍ PRO VYUŽITÍ PROGRAMOVÁNÍ PŘI ŘEŠENÍ MATEMATICKÝCH ÚLOH MO V PRAXI

Pokud by bylo využito programování jako prostředku pro řešení matematických úloh (nejen) MO řešitelných pomocí systematického experimentování, pak by bylo žádoucí splnit následující kroky:

- řádně vysvětlit žákům princip systematického prohledávání stavového prostoru;
- vysvětlit žákům nevhodnost použití metod *rand.randint()* či *rand.randrange()* pro účely vyčíslování řídících proměnných: pokusit se pro účely vyčíslení řídících proměnných využít *for* cykly, popř. *while* cyklus;
- vysvětlit žákům výhody a nevýhody získání jednotlivých cifer vícecifferných přirozených čísel pomocí přetypování přirozeného čísla *a* na řetězec pomocí *i = str(a)* a práce s jednotlivými ciframi pomocí indexace *i[0]*, *i[1]*, … apod.;
- porovnat se žáky přístupy v získávání jednotlivých cifer u vícecifferných přirozených čísel: porovnat přístup pomocí výše uvedeného přetypování přirozeného čísla na řetězec s přístupem realizace ciferného rozkladu pomocí celočíselného dělení a operace modulo: *//* a *%*;
- vysvětlit žákům nutnost či vhodnost použití logických spojek *and* a *or* při posuzování splnění dvou a více podmínek;
- vysvětlit žákům nevhodnost použití vzájemně vnořených podmíněných příkazů *if* u těch podmínek, které lze zformulovat pomocí za použití spojek *and* a *or* a menšího počtu podmíněných příkazů *if;*
- vysvětlit žákům specifika posuzování dílčích podmínek pomocí systému *if – continue*;

- vysvětlit žákům možnosti vhodného zúžení prohledávaného stavového, prostoru, např. zúžení na první polovinu původního stavového prostoru či zúžením stavového prostoru pomocí druhé odmocniny původní horní číselné meze;
- vysvětlit žákům výhody a nevýhody předčasného ukončení stavového prostoru s použitím *break*;
- zdůraznit žákům, že je nutné psát zdrojové kódy v souladu s doporučenými styly psaní zdrojových kódů v jazyce Python (PEP 8), které uvádí dokument [11].

6 ZÁVĚR

V předloženém článku jsme prezentovali výsledky realizované výzkumné sondy, která se zabývala empirickým ověřením pěti vybraných matematických úloh, které byly inspirovány matematickými úlohami MO a byly řešitelné pomocí systematického experimentování.

V úvodu článku vysvětlujeme podstatu systematického experimentování a uvádíme příležitosti a hrozby uplatnění programování při řešení matematických úloh MO. V metodologicky zaměřené kapitole komentujeme realizaci výzkumné sondy a tvorbu dotazníku vlastní konstrukce. Uvádíme zadání pěti testových úloh, příslušné zdrojové kódy funkčních programů v jazyce Python a příslušné výstupy programů. V kapitole, jíž obsahem je prezentace výsledků výzkumné sondy, uvádíme nejen postoje respondentů k vybraným faktorům řešení úloh, ale také prezentujeme algoritmické konstrukce, které respondenti během řešení úloh uplatnili. V závěru článku jsou formulována doporučení pro využití programování při řešení matematických úloh v praxi.

Lze konstatovat, že všechny formulované výzkumné otázky byly úspěšně zodpovězeny. V rámci první výzkumné otázky ve znění „Jaká je vnímaná snadnost pochopení zadání úloh, vnímaná snadnost vytvoření funkčních programů a míra inspirace řešení úloh z autorského řešení úloh v odborném článku?“, která zkoumala tři postojové složky respondentů v procesu řešení úlohy. První byla zaměřena na vnímání obtížnosti zadání úloh. V zadaných pěti úlohách byly vyjádřeny poměrně různorodé míry snadnosti pochopení zadání úloh (vzestupně: 50%, 72%, 74%, 84% a 88 %). Až na jednu úlohu lze považovat ve zbývajících čtyřech úlohách jejich zadání jako snadno pochopitelné. Druhá postojová složka se týkala vnímané míry snadnosti vyřešení úloh napsáním funkčních programů v jazyce Python. V zadaných pěti úlohách byly vyjádřeny poměrně různorodé míry snadnosti vyřešení úloh (vzestupně: 50%, 60%, 66%, 68% a 70%). Z těchto výsledků lze usuzovat, že respondenti klasifikovali snadnost vyřešení úloh, která by se dala interpretovat jako snadnost střední až středně vyšší. Poslední, třetí, postojová složka se týkala vyjádření míry inspirace řešení vzorově řešenými úlohami. V zadaných pěti úlohách byly vyjádřeny relativně stejné míry inspirace vzorově řešenými úlohami (vzestupně 18%, 20% 26%, 26% a 28%). Ze zjištění lze usuzovat, že se respondenti relativně málo inspirovali vzorově řešenými úlohami.

V rámci druhé výzkumné otázky ve znění „Jaká je míra úspěšnosti řešení jednotlivých úloh?“. Jako správně vyřešené úlohy byly uznány pouze zdrojové kódy u těch úloh, které vypisovaly zcela všechny správné výsledky. Úspěšnost vyřešení pěti zadaných úloh byla poměrně různorodá (vzestupně 38%, 44%, 50%, 58% a 72%). Tuto různorodost lze vysvětlit jak mírou úspěšnosti pochopení zadání úloh, tak také mírou obtížnosti tvorby algoritmických konstrukcí nutných pro vyřešení úloh.

V rámci třetí výzkumné otázky jsme analyzovali všechny prezentované strategie řešení úloh respondentů. Zajímaly nás všechny algoritmické konstrukce, které respondenti prezentovali v rámci svých řešení. Respondenti překvapili originalitou předložených zdrojových kódů. Například vyřešili jednu z úloh nikoliv systematickým prohledáváním stavového prostoru, ale nalezli vzorec pro vyčíslení hledaného řešení. Dále pak místo systematického prohledávání stavového prostoru prohledávali stavový prostor nesystematicky, pomocí náhodného vygenerování možných variant řešení s následným posouzením splnění podmínek takové varianty. V rámci práce s ciframi víceciferných čísel nevyužili číselné rozklady těchto čísel pro práci s jednotlivými ciframi těchto

čísel, ale uplatnili přístup ryze informatický: dané číslo přetypovali na řetězec, s ciframi pracovali jako se znaky tohoto řetězce a vytvořili nové víceciferné číslo s těchto znaků jako nový řetězec s následným přetypováním na celé číslo. Všechna taková zjištění byla v článku relativně podrobně popsána.

Z dosažených výsledků usuzujeme, že výsledky naší výzkumné sondy by mohly zaujmout odbornou veřejnost, zejména z řad učitelů informatiky, matematiky a didaktiků informatiky.

PODĚKOVÁNÍ

Touto cestou bychom rádi poděkovali panu Ing. Bc. Jaroslavu Semerádovi, MBA, řediteli Střední průmyslové školy a Vyšší odborné školy v Liberci, za podporu realizace výzkumného šetření. Mgr. Lukáši Tichému z Gymnázia Chomutov a Mgr. Jiřímu Fišerovi, Ph.D. z Přírodovědecké fakulty UJEP v Ústí nad Labem děkujeme za konstruktivní připomínky k rukopisu tohoto článku. Stejně tak děkujeme všem respondentům výzkumné sondy za jejich účast na dotazníkovém šetření.

BIBLIOGRAFICKÉ ODKAZY

[1] edu.cz. RVP ZV – Rámcový vzdělávací program pro základní vzdělávání. [online]. Dostupné z: https://edu.gov.cz/rvp-ramcove-vzdelavaci-programy/ramcovy-vzdelavacici-program-pro-zakladni-vzdelavani-rvp-zv/ [cit. 2025-02-11].

[2] edu.cz. RVP G – Rámcový vzdělávací program pro gymnázia. [online]. Dostupné z: https://edu.gov.cz/rvp-ramcove-vzdelavaci-programy/ramcove-vzdelavaci-programy-pro-gymnazia-rvp-g/ [cit. 2025-02-11].

[3] https://www.matematickaolympiada.cz/

[4] https://skmo.sk/

[5] PERK, L. Pojďme hledat řešení některých úloh matematické olympiády pomocí7 programování I. In: *Rozhledy matematicko-fyzikální.* [online]. 2024, 99(1). [cit. 2025-02-11]. https://rozhledy.jcmf.cz/wp-content/uploads/RMF-99-1.pdf

[6] PELIKÁN, J. *Základy empirického výzkumu pedagogických jevů*. Praha: Karolinum, 2011. ISBN 978-80-246-1916-3.

[7] POLYA, G. *Jak to řešit?: překvapivé aspekty (nejen) matematických metod*. Praha: MatfyzPress, 2016. ISBN 978-80-7378-325-9.

[8] EISENMANN, P., PŘIBYL, J. Systematické experimentování ve výuce matematiky. In: *Sborník příspěvků 6. konference Užití počítačů ve výuce matematiky.* České Budějovice: Jihočeská univerzita v Českých Budějovicích, 2013. [online]. [cit. 2025-02-11]. https://home.pf.jcu.cz/~upvvm/2013/sbornik/clanky/10_UPVM2013_Eisenmann_Pribyl.pdf

[9] HVORECKÝ, J. Riešenie matematických úloh hrubou silou. In: *Dva dni s didaktikou matematiky 2022: Zborník príspevkov*. Bratislava: Univerzita Komenského v Bratislavě, 2022. [online]. [cit. 2025-02-11]. https://www.comae.sk/zbornik2022.pdf

[10] KOPKA, J. Umění řešit matematické problémy. Praha: HAV, 2013. ISBN 978-80-903625-5-0.

[11] https://peps.python.org/pep-0008/

Konečný automat a Turingov stroj v úlohách súťaže iBobor

Finite automata and Turing machine in iBobor competition tasks

Monika Tomcsányiová
FMFI Univerzity Komenského v Bratislave
monika.tomcsanyiova@fmph.uniba.sk

EXTENDED ABSTRACT

In the school year 2024/25, the eighteenth edition of the Informatický bobor, abbreviated as iBobor, competition was held in Slovakia. The competition is known internationally as Bebras and involves more than seventy countries from all over the world. The tasks in the competition are selected by experts in the field of computer science and education. The text of the tasks is carefully and purposefully formulated for different age categories of students from 8 to 18 years old. The tasks include a variety of computer science concepts, focus on the use of algorithms and digital technologies, and are created with context and motivation appropriate to the age of the student. Data on the success rate of each task is available in EduPage. For our research, tasks containing a computational model of a finite automaton and a Turing machine were selected. In terms of the theoretical aspect of these abstract models, these topics are included in university courses.

The computational model of a finite automata and a Turing machine is used in different contexts, or with different motivations, quite regularly in competition problems. For tasks intended for lower age categories, a specific and familiar context is designed by the authors to be close to the pupils of the respective age category. Motivations of flowers, stringing beads, or analysing and designing patterns are often used in the tasks. For the higher age categories, there is an attempt to replace the symbolic notation of the Turing machine with less abstract models, such as recolouring squares according to predetermined rules.

In the Slovak version of the competition in the 2008/09 school year, the problem with the concept of finite automata was used for the first time for the Senior category. In subsequent editions, similar types of problems were also proposed for the lower competition categories. From the comparison of two competition tasks, Mother's Day and Bracelets, focused on the finite automaton model, our research identified six categories that could influence the difficulty of the task. After analyzing these categories in each task, no direct correlation was found between any of the proposed categories and task success. However, it was found that pupils aged 14 to 15 are able to solve this type of problem with success rates of up to 82% (Mother's Day). Problems with a finite automaton context will be included in the iBobor competition in future years and further research will investigate pupils' success in solving them.

Only one task containing a graphical representation of a computational model of a Turing machine has been used in the Slovak competition so far, the task Robot, or Colorful Robot. The task was included for pupils aged 14 to 18 years (Cadet, Junior and Senior). In our research, several significant factors were identified that may have influenced the difficulty of this task. The research showed that tasks of this type are difficult for pupils aged 14 to 15 years (success rate only 22%), but pupils aged 17 to 19 years can grasp this computer science concept quite well. Their success rate was 42%, which is the expected success rate for a difficult competitive assignment. Based on our research, tasks with the concept of a Turing machine will be included only for the oldest students.

Keywords

iBobor competition, finite automata, Turing machine, computer science

ABSTRAKT

V školskom roku 2024/25 prebehol už 18. ročník súťaže Informatický bobor (iBobor). Úlohy z tohto, ale aj z minulých ročníkov sú žiakom a učiteľom dostupné po celý rok v systéme EduPage. Žiaci ich môžu riešiť kedykoľvek, či už na počítačoch alebo mobilných zariadeniach. Učiteľ môže zadania využívať na hodinách informatiky alebo v rámci domácich zadaní. Výhodou súťažných úloh je, že text ich zadania je krátky a dajú sa vyriešiť za niekoľko málo minút. Úlohy pritom obsahujú rôzne informatické koncepty a sú zamerané na použitie algoritmov a digitálnych technológií pri riešení zadaní s motiváciou vhodnou pre príslušný vek žiaka. Pre náš výskum sme si vybrali skupinu úloh, ktorých spoločným menovateľom je výpočtový model konečného automatu a Turingovho stroja. Ak uvažujeme o teoretickej stránke týchto konceptov je zrejmé, že kvôli silnej abstrakcii sa tieto témy učia až na vysokej škole. Niektoré súťažné úlohy s týmito konceptami však dokazujú, že grafickému významu konečných automatov dokážu do istej miery porozumieť aj žiaci základnej školy. A naopak, zistili sme, že niektoré z konceptov sú pre žiakov na základnej škole príliš náročné. V článku budeme skúmať reprezentácie konečného automatu a Turingovho stroja a úspešnosť žiakov pri riešení týchto dvoch typov úloh.

Kľúčové slová

súťaž iBobor, konečný automat, Turingov stroj, informatika

1 ÚVOD

Súťaž Informatický bobor (iBobor), medzinárodne nazývaná Bebras, bola prvýkrát organizovaná v Litve v roku 2004 a odvtedy sa rozšírila do viac ako 85 krajín [1]. Jej cieľom je zvýšiť záujem žiakov vo veku 8 až 18 rokov o digitálne technológie a o riešenie úloh s rôznymi informatickými konceptmi. Súťaž kladie dôraz na pochopenie základných pojmov informatiky pred technickými detailmi, faktami a pojmami. Súťaž je implementovaná samostatne a každá zúčastnená krajina navrhuje svoju skupinu úloh, ktorú žiaci riešia. Národné odlišnosti v jej realizácii sú uvedené v článkoch, pozri napr. [2], [3]. Súťažiaci na Slovensku riešia online na počítači počas 30 až 40 minút 9 až 15 úloh rôznej náročnosti, pozri [4]. Žiaci odpovedajú priamo vo webovom prehliadači rôznym spôsobom: výberom z viacerých možností, interakciou alebo vpísaním odpovede do vstupného textového poľa vpísaním číslic alebo znakov. Zadania sú pripravované tak, aby sa na ne dalo odpovedať bez predchádzajúcich znalostí informatického konceptu, ktorý sa nachádza v zadaní. Úloha často obsahuje izolované modely súvisiace s jedným, alebo aj viacerými informatickými konceptami. Tieto informatické koncepty sú žiakom predkladané prostredníctvom príbehov a obsahujú vizuálne a interaktívne prvky, ktoré už na prvý pohľad zaujmú študentov.

Viaceré štúdie a články z Bebras komunity sa zameriavajú na rôzne aspekty súťažných úloh. Článok [5] skúma problémy úloh súťaže a zdôrazňuje, že zadania by mali byť pútavé pre žiakov, primerané veku, náročné, ale riešiteľné, s dôrazom na informatické koncepty v nich použité. V našom výskume skúmame úspešnosť úloh, v ktorých sa vyskytuje výpočtový model konečného automatu a Turingovho stroja.

Výskumom chápania konečných automatov v základoškolskom a stredoškolskom vzdelávaní sa zaoberá článok [6]. Na oboznámenie žiakov s konečným automatom používajú učenie založené na hádankách. Navrhli hru, v ktorej automat vyrába rôzne predmety a úlohou žiakov je, aby zbierali „dobré“ predmety. Žiaci tak môžu zažiť základný koncept automatov bez formálnych definícií. Ich štúdia ukázala, že niektorí žiaci sú schopní pochopiť tento pojem, ale väčšina žiakov mu dokáže porozumieť úplne a až 80 % nedokázalo zistiť vlastnosti automatov v otázkach rozpoznávania predmetov podľa zadaných diagramov. V článku [7] popisujú ako žiaci dostali model päťpáskového Turingovho stroja, ktorý používali na návrh vlastnej hry. Štúdia ukázala, že niektorí zo žiakov využívajú prístup tvorbou vlastnej hry, ale iní uprednostňujú tradičnejšie metódy výučby.

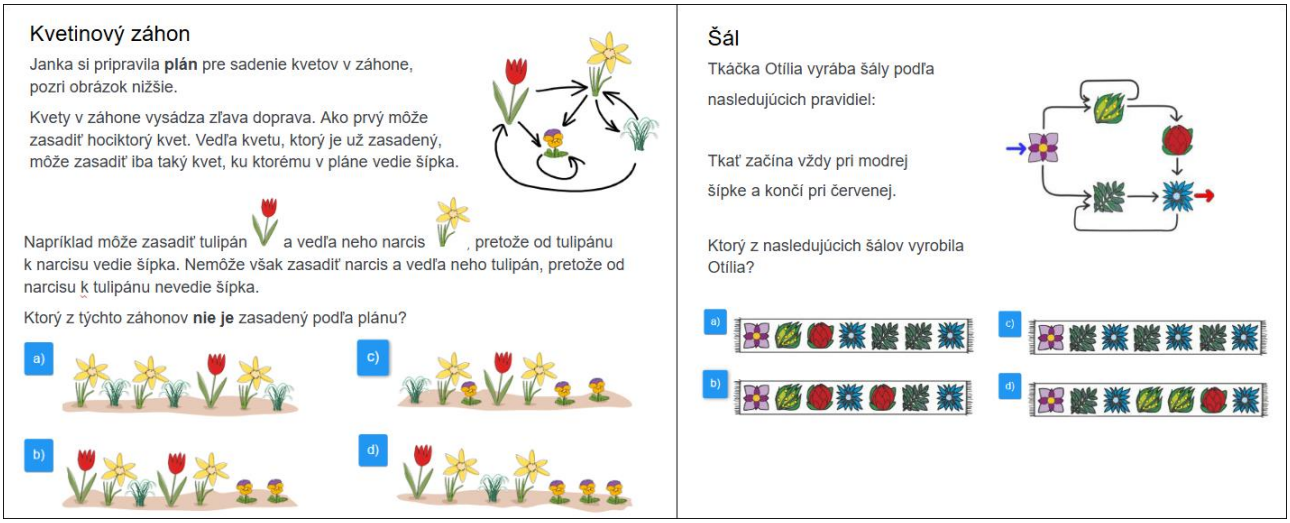
Pri zavádzaní nového ponímania informatiky na Novom Zélande, pozri [8], si uvedomovali, že keď učitelia uvidia témy ako „formálne jazyky a automaty“ môžu mať pochybnosti, či autori nového kurikula nezašli príliš ďaleko. Preto vysvetľovali učiteľom, že účelom nového učebného plánu chceli dať možnosť žiakom nazrieť do niektorých oblastí informatiky. Žiaci by mali mať možnosť stretnúť sa s týmito konceptami bez toho, aby sa museli zaoberať ich teóriou. Podobný prístup je aplikovaný aj úlohách súťaže iBobor.

2 KONEČNÝ AUTOMAT A TURINGOV STROJ V INFORMATIKE

V odbornej informatickej literatúre nájdeme definície konečného automatu pomocou množiny stavov a prechodových funkcií. Je zrejmé, že takýto zápis nie je možné prezentovať žiakom základnej a ani strednej školy. Konečné automaty sa však svojím grafickým vyjadrením podobajú na ohodnotené grafy a môžu byť reprezentované pomocou diagramu s vrcholmi a hranami. Komunita informatických odborníkov pripravujúcich súťaž Bebras, pozri [1], navrhuje a pripravuje takýto typ úloh. V odporúčaných úlohách sa takéto zadania objavujú pre vekové kategórie od 8 až po 18 rokov. V slovenskej verzii súťaže bola v školskom roku 2008/09 prvýkrát použitá úloha s konceptom konečných automatov pre kategóriu Senior. V ďalších ročníkoch boli takéto úlohy navrhnuté aj do nižších súťažných kategórií. Úloha obsahujúca grafické znázornenie výpočtového modelu Turingovho stroja bola v slovenskej súťaži zatiaľ použitá iba jedna, zadanie Robot, resp. Farebný robot, pozri Obrázok 5. Úloha bola v školskom roku 2017/18 zaradená do troch súťažných kategórií. Skúmanie úspešnosti žiakov a hľadanie možných príčin rozdielov pri riešení týchto úloh je cieľom nášho prieskumu.

3 ÚLOHY V SÚŤAŽI

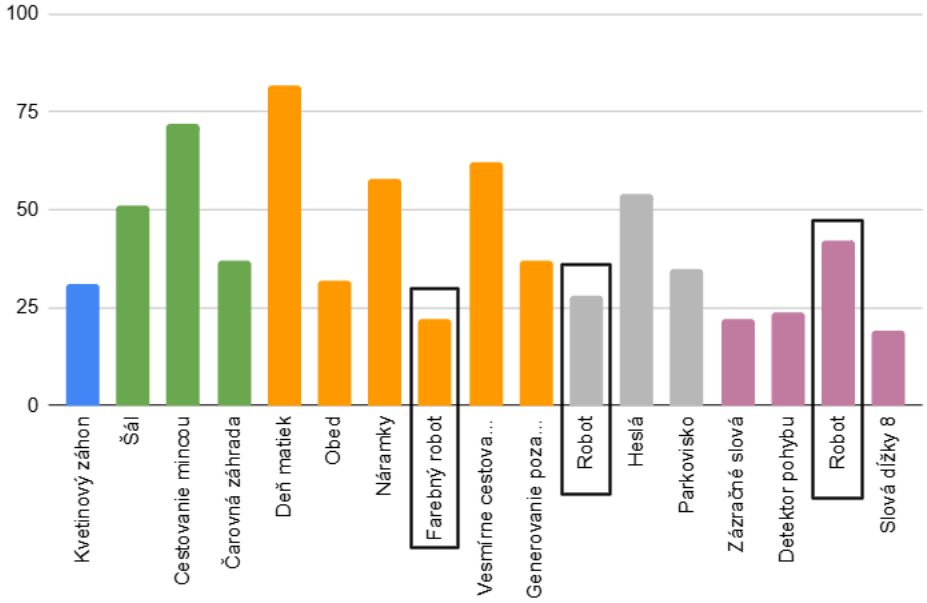
Výpočtový model konečného automatu a Turingovho stroja sa v rôznych kontextoch, resp. s odlišnými motiváciami, objavuje v súťažných úlohách pomerne pravidelne. Pre úlohy určené pre nižšie vekové kategórie sa autori snažia navrhovať konkrétny a známy kontext tak, aby bol blízky žiakom príslušnej vekovej kategórie. V úlohách sú často používané motivácie kvetov, navliekania korálikov, či navrhovanie vzorov, pozri Obrázok 1. Pre vyššie vekové kategórie sa autori snažia o motiváciu, ktorá by nahradila symbolický zápis inými, menej abstraktnými, modelmi. Pre Turingov stroj bolo v úlohe Robot zvolené farebné grafické znázornenie s obrázkom štvorca, pozri Obrázok 5 v časti 3.2.



Obrázok 1. Vľavo úloha kategórie Bobrík, vpravo zadanie úlohy z kategórie Benjamín.

V rámci nášho prieskumu sme zisťovali, aké zadania s témou konečného automatu a Turingovho stroja boli v národnej slovenskej verzii súťaže Bebras použité od školského roku 2013/14 po školský rok 2024/25 pre všetky súťažné kategórie. Podrobným prieskumom viac ako tisíc úloh, ktoré sú v súčasnosti v archíve iBobor sme zistili, že úloh, ktoré obsahujú koncept konečného automatu alebo

Turingovho stroja bolo sedemnásť. Bližším skúmaním sme zistili, že v každom ročníku súťaže, okrem ročníkov 2018/19 a 2021/22, bola aspoň v jednej kategórii použitá úloha s konceptom konečného automatu.

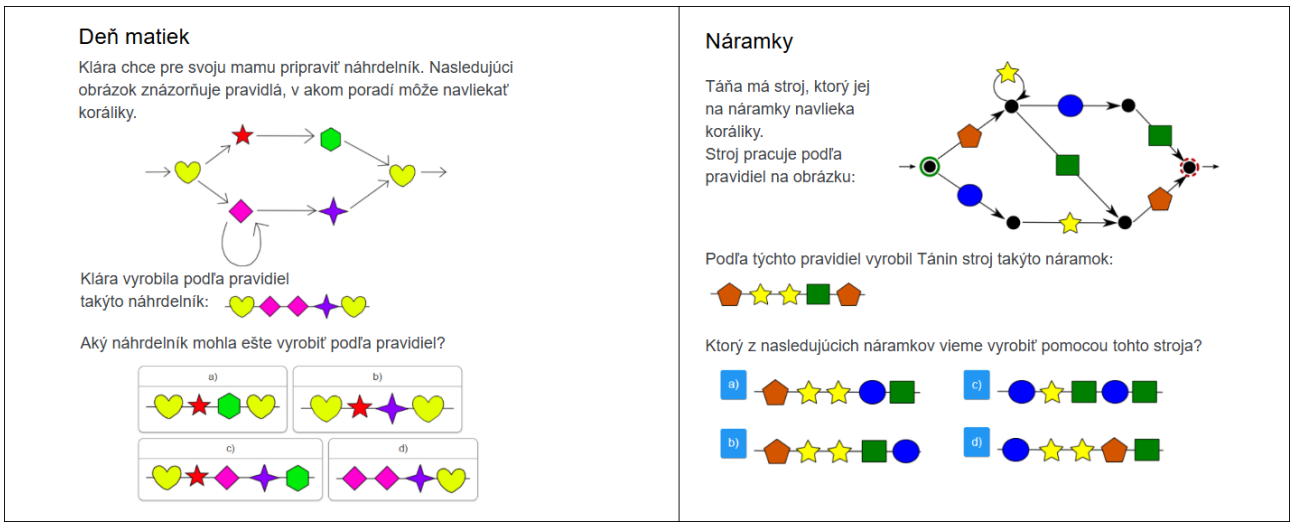


Obrázok 2: Graf úspešnosti riešenia úloh s farebne odlíšenými kategóriami (Bobrík, Benjamín, Kadet, Junior a Senior)

Najviac, šesť úloh, bolo zaradených do kategórie Kadet. Zaujímavé je, že konečný automat obsahovalo aj jedno zadanie pre kategóriu Bobrík, teda pre žiakov 4. a 5. ročníka základnej školy, pozri úlohu Kvetinový záhon na Obrázku 1 vľavo. Graf na Obrázku 2 vyjadruje úspešnosť týchto úloh pre jednotlivé kategórie. Zvislá os určuje percento správnych odpovedí. Farby stĺpcov určujú jednotlivé súťažné kategórie v poradí Bobrík, Benjamín, Kadet, Junior a Senior, v ktorých sa tieto zadania objavili. V čiernom obdĺžniku je úloha Robot obsahujúca model Turingovho stroja, ktorá bola v troch kategóriách. Na prvý pohľad vidíme, že úspešnosť týchto úloh je pomerne nízka. Zároveň však vidíme, že úspešnosť žiakov pri riešení úloh kolíše. Niektoré úlohy majú vysokú úspešnosť, dokonca viac ako 80 %, na rozdiel od zadaní s úspešnosťou iba okolo 25 %. Aké môžu byť príčiny tohto javu?

3.1 Konečný automat v kategórii Kadet

Súťažná kategória s názvom Kadet je určená pre žiakov 8. a 9. ročníka základnej školy, resp. terciu a kvartu osemročného gymnázia. Každoročne sa jej zúčastňuje viac ako desaťtisíc žiakov, pozri [4]. Pri skúmaní zadaní obsahujúcich konečný automat, sme zistili, že úloha s týmto konceptom bola pre kategóriu Kadet prvýkrát zaradená v školskom roku 2013/14, pozri zadanie Deň matiek na Obrázku 3 vľavo. Súťažilo 13 794 žiakov a v grafe na Obrázku 2 vidíme, že úspešnosť žiakov pri jej riešení bola 82 %, čo je v rámci očakávania pre ľahké súťažné zadanie Ako motivácia, resp. kontext, úlohy bol použitý náhrdelník, na ktorý sa koráliky navliekajú podľa určitých pravidiel. Pravidlá navliekania boli určené diagramom.



Obrázok 3: Dve na prvý pohľad veľmi podobné zadania pre kategóriu Kadet

Zadanie úlohy Deň matiek, bez podrobného vysvetľovania konceptu konečného automatu, zobrazilo žiakom stavový diagram. Stavy sú zobrazené ako koráliky, ktoré sa navliekajú do náhrdelníka. Šípky určujú, ktoré koráliky môžu nasledovať za sebou. V diagrame bol použitý, opäť bez akéhokoľvek vysvetlenia, aj koncept viacnásobného navliekania toho istého korálika za sebou. Ďalej bol v texte úlohy uvedený príklad jedného náhrdelníka, ktorý môže vzniknúť podľa diagramu. Príklady, ktoré sú uvádzané v úlohách súťaže, by mali žiakom umožniť lepšie pochopenie zadania a ukazovať možné situácie predovšetkým s takými príkladmi, ktoré sú založené na očakávaných miskoncepciách. V tomto prípade je uvedený teda príklad správne vytvoreného náhrdelníka, ktorý zámerne ukazuje, že ružový diamant je možné použiť viackrát za sebou. Úlohou žiakov pri riešení úlohy bolo určiť, ktorý ďalší náhrdelník, uvedený v jednotlivých distraktoroch, môže byť tiež vyrobený podľa diagramu. Táto úloha bola zaradená pre ako ľahká, teda autori predpokladali, že bude mať úspešnosť viac ako 70 %. Výsledky, ktoré žiaci pri jej riešení dosiahli, teda 82 % správnych odpovedí, potvrdili, že žiaci tento typ diagramu pochopili a úlohu dokázali vyriešiť s očakávanou úspešnosťou pre ľahké súťažné zadanie.

Podľa úspešnosti žiakov kategórie Kadet pri riešení tejto úlohy, bola medzi autormi úlohy prijatá hypotéza, že takéto úlohy sú pre túto vekovú kategóriu vhodné ako ľahké zadania. V školskom roku 2016/17, keď v kategórii Kadet súťažilo 14 347 žiakov, bola sú súťaže zaradená úloha Náramky, pozri Obrázok 3 vpravo. Úspešnosť riešenia úlohy Náramky však bola len 58 %. Aké môžu byť príčiny toho, že na prvý pohľad takmer identické úlohy majú tak rozdielnu úspešnosť?

Po vizuálnej stránke sú si úlohy Deň matiek a Náramky veľmi podobné. Z pohľadu konečných automatov bola v úlohe Náramky zmenená grafická stránka zadania. V grafickom znázornení konečného automatu bol použitý význam stavov a prechodových funkcií viac podobný na zápis z teoretickej informatiky. Stavy v tomto zobrazení nie sú pomenované, šípkami a zelenou, resp. červenou farbou, je naznačený počiatočný a koncový stav. Prechodové funkcie sú určené šípkami, na ktorých sú nakreslené výrazne farebne aj typovo odlišné koráliky (hviezda, štvorec, kruh, päťuholník). Zadanie úlohy, podobne ako v úlohe Deň matiek, obsahuje príklad, ktorý zobrazuje konkrétny náramok, ktorý sa dá vyrobiť podľa uvedeného diagramu. Príklad zároveň objasňuje možnosť viacnásobného použitia hviezdičky v náramku. Úlohou žiakov bolo, rovnako ako v úlohe Deň matiek, označiť ďalší z náramkov, ktoré je možné vyrobiť.

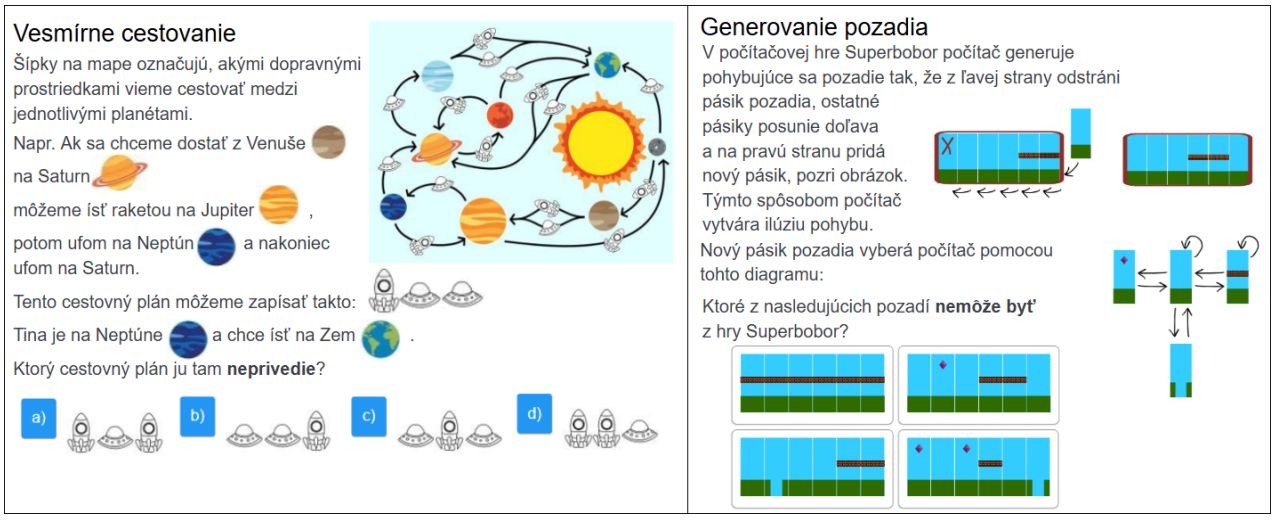
Z porovnania úloh Deň matiek a Náramky sme určili šesť kategórií, pozri Tabuľku 1, ktoré budeme skúmať aj pre ďalšie úlohy obsahujúce konečný automat. Tabuľku sme rozšírili o úlohy Vesmírne cestovanie a Generovanie pozadia, pozri Obrázok 4. Pre úplnosť sme do tabuľky zahrnuli aj počet súťažiacich a školský rok (tieto hodnoty uvádzame ako doplnkovú informáciu, nejedná sa o skúmané kategórie). V tabuľke sú úlohy zoradené podľa roku ich zaradenia do súťaže.

Tabuľka 1: Kategórie, ktoré by mohli mať vplyv na úspešnosť úloh s konečným automatom

	kontext, resp. motivácia	stavy	prechodová funkcia	viacnásobné prechádzanie medzi dvoma stavmi	dĺžka	druh odpovede	počet súťažiacich / školský rok	úspešnosť úlohy
Deň matiek	jednoduché farebné geometrické útvary, dobre navzájom rozlíšiteľné, rozdielne na prvý pohľad (srdce, hviezda...)	obrázky ako koráliky	šípka	nie	4 až 5	kladná	13 794 2013/14	82 %
Náramky	jednoduché farebné geometrické útvary, dobre navzájom rozlíšiteľné, rozdielne na prvý pohľad (štvorec, kruh...)	čierne kruhy s vyznačením počiatočného a koncového stavu	šípka s korálikom	nie	5	kladná	14 347 2016/17	58 %
Vesmírne cestovanie	cestovanie raketami vo vesmíre z planéty na planétu	obrázky planét, farebné, pomerne dobre rozlíšiteľné	šípka s obrázkom vesmírnych rakiet	áno	3	záporná	17 530 2019/20	62 %
Generovanie pozadia	generovanie pozadia v počítačovej hre, farebné pásiky, rozdiely iba v detailoch (hnedý pásik, diamant)	obrázky pásikov generovaného pozadia	pomerne malá nevýrazná šípka	áno	6	záporná	24 217 2024/25	37 %

Ani po analýze týchto kategórií sa nám nepodarilo vyšpecifikovať hlavnú príčinu, ktorá by mohla viesť k pomerne výraznej odlišnosti v úspešnosti žiakov pri riešení týchto úloh. Hoci mala úloha Vesmírne cestovanie zápornú otázku (čo sa pri úlohách bežne považuje za náročnejší typ otázky), mala o 4 %vyššie percento úspešnosti ako úloha Náramky.

Rozhodli sme sa tiež preskúmať úspešnosť týchto úloh v systéme EduPage [9]. Počas uplynulých štyroch rokov boli do systému prenesené takmer všetky úlohy z minulých ročníkov súťaže iBobor. Tieto úlohy sú v systéme trvalo dostupné pre učiteľov a ich žiakov. Žiak si môže hocikedy zvoliť, že chce riešiť úlohy zo súťaže. Môže si vybrať, že chce riešiť príslušnú sadu úloh, ktoré boli v daný školský rok v súťaži. Alebo úlohu nájde medzi sadami podobne náročných úloh, ktoré sú uložené v systéme. Učiteľ môže počas celého školského roka určovať, ktoré úlohy majú jeho žiaci riešiť počas vyučovacej hodiny alebo v rámci domácej prípravy žiakov na vyučovanie. V oboch prípadoch systém eviduje početnosť a úspešnosť riešenia každej úlohy.



Obrázok 4: Úlohy Vesmírne cestovania a Generovanie pozadia z kategórie Kadet.

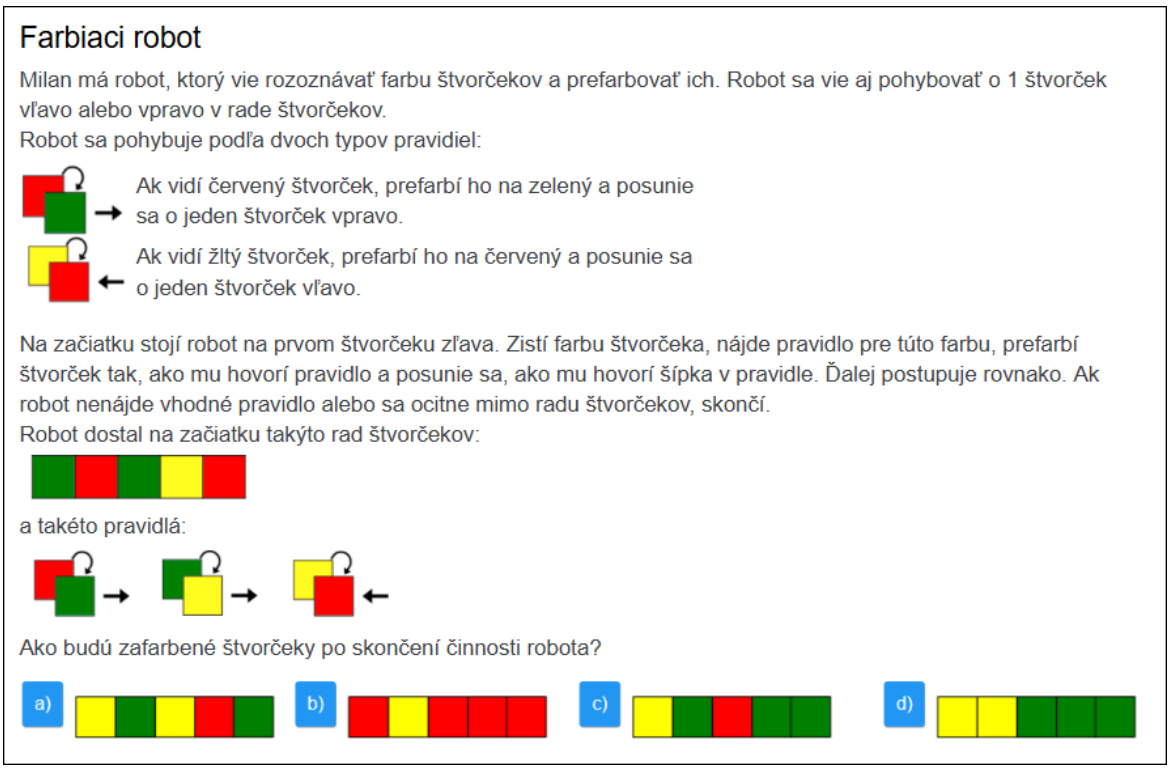
V systéme EduPage sme skúmali štyri vyššie uvedené úlohy. Zamerali sme sa iba na žiakov, ktorí majú v systéme uvedený ôsmy a vyšší ročník, teda na žiakov, ktorí patria do kategórie Kadet a pre ktorých boli tieto úlohy určené v súťaži. Úlohu Deň matiek riešilo v EduPage 193 Kadetov a má úspešnosť 59 %, Náramky riešilo 891 Kadetov a má úspešnosť 55 %, Vesmírne cestovanie riešilo 2664 Kadetov a správne ju vyriešilo 44 % z nich, Generovanie pozadia v systéme po ukončení súťaže v školskom roku 2024/25 riešilo 214 Kadetov, správne ju vyriešilo 38 % žiakov. Z tejto štatistiky vidíme napríklad, že úspešnosť úloh Deň matiek a Náramky, ktoré mali v súťaži rozdiel 24 % je v systéme iba 4 %. Ale aj to, že hoci mala úloha Vesmírne cestovanie počas súťaže úspešnosť až 62 %, v systéme ju správne riešilo iba 44 % žiakov. Na prvý pohľad sa zdá, že žiaci riešia tieto úlohy aj v rámci domácej úlohy alebo v prípade, že si úlohu vyberú na riešenie sami, s veľmi rozdielnou úspešnosťou.

3.2 Turingov stroj pre kategórie Kadet, Junior a Senior

Turingov stroj je zovšeobecnenie konečného automatu. Keďže Turingov stroj pracuje na nekonečnej páske a obsahuje hlavu pohybujúcu sa oboma smermi, ktorou číta a aj zapisuje symboly na pásku, nie je preň možné nakresliť statický diagram tak, ako to bolo pre konečný automat. V školskom roku 2017/18 v úlohách Farbiaci robot, resp. Robot pre kategórie Junior a Senior, bol namiesto symbolov známych z teoretickej informatiky, použitý obrázok farebných štvorcov, pozri Obrázok 5. Autori úloh usúdili, že tento symbol by mohol byť pre žiakov vo veku 14 až 18 rokov pochopiteľný a tiež verili, že žiaci túto abstrakciu dokážu počas súťaže pochopiť. Abecedou prezentovaného Turingovho stroja sú rôznofarebné štvorce a prechodová funkcia je určená tým, že je pomocou šípky zobrazená zmena farby štvorca na základe farby, na ktorej sa nachádza hlava. Posun hlavy môže byť smerom doprava alebo doľava, pozri Obrázok 5.

Prefarbovanie štvorcov na základe daných pravidiel sa ukázalo ako náročný koncept. V kategórii Kadet súťažilo 14 687 žiakov, ich úspešnosť pri riešení tejto úlohy bola iba 22 %. Pre kategórie Junior a Senior bolo navrhnuté mierne odlišné zadanie, ktoré obsahovalo náročnejšiu prechodovú funkciu a dlhší vstup. V kategórii Junior bolo 7 775 súťažiacich, ich úspešnosť bola 28 %. Seniorov súťažilo 4 716, úlohu vyriešilo správne 42 % z nich.

Zadanie úlohy Farbiaci robot bolo zaradené do troch súťažných kategórií ako ťažké. Očakávaná úspešnosť pre päť úloh, ktoré sú pre kategórie nasadené ako ťažké je okolo 30 až 50 percent. Podľa dosiahnutých výsledkov bola očakávaná úspešnosť úlohy dosiahnutá iba v kategórii Senior.



Obrázok 5: Zadanie úlohy Farbiaci robot pre kategóriu Kadet.

Pozrime sa na možné príčiny nízkej úspešnosti tejto úlohy:

- Zadanie obsahuje pomerne veľa textu a pre žiakov, ktorí pred súťažou nepoznali tento výpočtový model, mohlo byť náročné pochopiť fungovanie Turingovho stroja počas riešenia úlohy v krátkom čase súťaže. Pri čase 40 minút, ktorý majú žiaci na 15 úloh, vychádza na jednu úlohu iba cca 3 až 5 minút. Tento čas však závisí od toho, v akom čase sa žiakovi podarí vyriešiť ostatné úlohy.
- Znázornenie prechodovej funkcie, a tiež pohybu hlavy, je schematicky znázornené pomocou farebných štvorcov a šípok. Obrázkom šípky je však naznačený aj smer pohybu hlavy stroja. Použitie šípok s podobným dizajnom, ale s rôznym významom mohlo byť pre žiakov mätúce.
- V zadaní nie je uvedený ukážkový príklad, ktorý by pomohol žiakovi ujasniť si, či z nákresu pochopil fungovanie stroja.
- Ukončenie činnosti stroja bolo definované v texte výrokom: „Ak robot nenájde vhodné pravidlo alebo sa ocitne mimo radu štvorčekov, skončí.“ Táto veta obsahuje sloveso v zápore "nenájde" a spojku "alebo". Takto formulované pravidlo mohlo byť pre žiakov náročné a nedokázali ho správne aplikovať.
- Zistenie správnej odpovede vyžadovalo od žiakov simuláciu prechodovej funkcie na vstupnej farebnej postupnosti. Bez precízneho zapisovania si vznikajúcej postupnosti bolo náročné zistiť správny výsledok aj pre krátku vstupnú postupnosť farieb.
- Aj v prípade pochopenia princípu stroja je možné, že žiaci urobili chybu pri samotnej simulácii a dostali tak nesprávny výsledok.

4 ZÁVER

V našom výskume sme sa zamerali na úlohy obsahujúce výpočtový model konečného automatu a Turingovho stroja v úlohách súťaže iBobor. Zistili sme, že úlohy s týmito konceptami sú pomerne pravidelne zaraďované do súťaže do všetkých súťažných kategórií okrem najmladších žiakov v kategórii Bobrík. Po zistení štatistických výsledkov pre všetky kategórie, pozri Obrázok 2, sme sa zamerali na štyri úlohy pre žiakov z kategórie Kadet. Pre úlohy obsahujúce konečný automat sme navrhli šesť kategórií, ktoré by mohli ovplyvňovať náročnosť úlohy, pozri Tabuľku 1 a skúmali sme

úspešnosť týchto úloh. Žiaľ, nepodarilo sa nám nájsť priamu koreláciu medzi žiadnou z navrhnutých kategórií a úspešnosťou úlohy. Keďže sa však ukázalo, že v niektorých prípadoch sú žiaci vo veku 14 až 15 rokov schopní riešiť tieto úlohy s úspešnosťou až 82 % (úloha Deň matiek), budeme tieto úlohy aj naďalej zaraďovať do našej národnej súťaže. Veríme, že ďalšie ročníky, do ktorých budú zaradené úlohy s konceptom konečného automatu, nám objasnia kolísanie úspešnosti žiakov pri ich riešení. V druhej časti sme skúmali úlohu s Turingovým strojom a úspešnosť jej riešenia v súťažných kategóriách Kadet, Junior a Senior. Ukázalo sa, že zadania tohto typu sú pre Kadetov, teda pre žiakov vo veku 14 až 15 rokov, náročné (úspešnosť iba 22 %). Výskum však ukázal, že Seniori, teda žiaci vo veku 17 až 19 rokov, tento informatický koncept dokážu v rámci súťaže pochopiť. Úspešnosť Seniorov pri riešení bola 42 %, čo je očakávaná úspešnosť pre ťažké súťažené zadanie. Výskum teda potvrdil, aj keď zatiaľ iba na jednej súťažnej úlohe, že zadania s konceptom Turingovho stroja môžeme zaraďovať do súťažnej kategórie Seniori.

5 BIBLIOGRAFICKÉ ODKAZY

[1] Bebras – International Challenge on Informatics and Computational Thinking. Dostupné z: http://www.bebras.org. [cit. 10.2.2025]

[2] Pluhár, Zs., Kaarto, H., Parviainen, M., Garcha, S., Shah, V., Dagienė, V., Laakso, M. J.: Bebras Challenge in a Learning Analytics Enriched Environment: Hungarian and Indian Cases. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives 2022, (s. 40-53). Springer, Cham.

[3] Vaníček, J., Šimandl, V., Dobiáš, V.: Bebras Tasks Based on Assembling Programming Code. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives 2022 (s. 113-124). Springer, Cham.

[4] iBobor – slovenská súťaž Informatický bobor. Dostupné z: http://ibobor.sk, [cit. 10.2.2025]

[5] Datzko, C., Datzko, S.: Aspects of designing a successful bebras challenge. In: Conference ISSEP 2021 Online Local Proceedings. Dostupné z: https://www.ru.nl/publish/pages/1026345/datzko_aspects_of_designing_a_successful_bebras_challenge.pdf. [cit. 10.2.2025]

[6] Isayama, D., Ishiyama, M., Relator, R., Yamazaki, K.: Computer Science Education for Primary and Lower Secondary School Students: Teaching the Concept of Automata. ACM Transactions on Computing Education, 17(1)(2), 2017.

[7] Korte, L., Anderson, S., Pain, H., Good, J.: Learning by game-building: A novel approach to theoretical computer science education. SIGCSE Bulletin 39, 3 (jun 2007), 53–57. DOI:http://dx.doi.org/10.1145/1269900.1268802

[8] Bell, T.:Establishing a nationwide CS curriculum in New Zealand high schools: providing students, teachers, and parents with a better understanding of computer science and programming. Communications of the ACM, 2014, 57(2), 28–30.

[9] EduPage – systém firmy aSc agenda. Dostupné z: https://www.edupage.org/ [cit. 10.2.2025]

Title:	**New Perspectives in Informatics Education – International Proceedings on Teaching Informatics**
Editors:	Mgr. Adam Dudáš, PhD., RNDr. Alžbeta Michalíková, PhD., doc. PaedDr. Patrik Voštinár, PhD., doc. Ing. Jarmila Škrinárová, PhD.
Edition:	First edition
Publisher:	Matej Bel University, Faculty of Natural Sciences in Banská Bystrica, Slovakia
Year:	2025
Pages:	74
Format:	electronic, conference proceedings
ISBN:	978-80-557-2249-8
EAN:	978055722498
DOI:	10.24040/2025.9788055722498