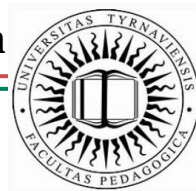
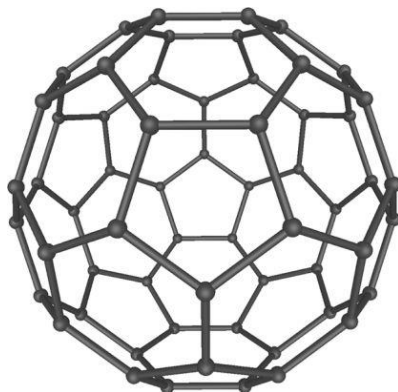




Trnava University in Trnava



Faculty of Education



XXXVIIITH DIDMATTECH 2025

**New methods and technologies in education, research
and practice**

Proceedings

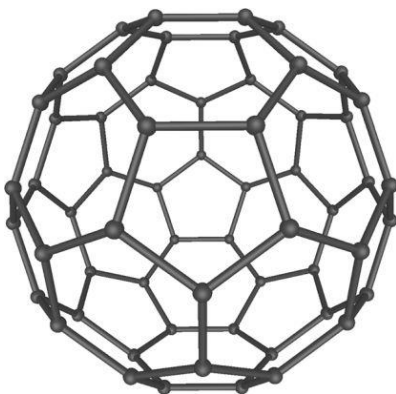
2025

ZBORNÍK
konferencie XXXVIII DidMatTech 2025

Proceedings of
XXXVIII DidMatTech 2025 Conference

*New Methods and Technologies in Education,
Research and Practice*

Trnava University in Trnava



2025

Editors: Prof. Ing. Veronika Stoffová, CSc.
PaedDr. Krisztina Czakóová, PhD.
Mgr. Ing. Roman Horváth, PhD.

Graphic editors:

© Editors and authors of papers

For the content of contributions are responsible their authors.

The contributions have not undergone editorial and linguistic corrections.

Za obsah jednotlivých abstraktov zodpovedajú ich autori.

Abstrakty neprešli redakčnou a jazykovou úpravou.

ISBN: 978-80-568-0796-5

EAN: 9788056807965

Reviewers – Posudzovatelia

The proceedings as a whole were reviewed by/ The proceedings as a publication item were assessed by: - Zborník ako publikačný celok posudzovali:

Prof. RNDr. Zuzana Hlaváčová, CSc.

doc. RNDr. Gabriela Lovászová, PhD.

doc. Ing. Melánia Feszterová, PhD.

Ing. Juraj Czifra, CSc.

Reviewers of individual articles

Posudzovatelia jednotlivých článkov

Krisztina Czakóová J. Selye University, Komárno

Veronika Stoffová Trnava University in Trnava

Ing. Ondrej Takáč, PhD. J. Selye University, Komárno

Prof. Kateryna Osadcha, DSc.

dr. Blanka Láng, PhD.

Dr. Csaba Holló, PhD. University of Szeged,

RNDr. József Udvaros, PhD University of Economics and Business,
Budapest

Ádám Gulácsi

Content – Obsah

Introduction – Úvod	9
Artificial intelligence and new challenges in education Umelá inteligencia a nové výzvy v edukácii	
1. Péter Antal: The New Challenges of Digital Education in Practice – A digitális oktatás új kihívásai a gyakorlatban (plenary presentation – invited speaker)	13
2. Ondrej Takáč, Kocsis Gergely: Possibilities of Close-up Photogrammetry in the Protection of Regional Cultural Heritage (plenary presentation – invited speakers)	31
3. Júlia Šebejová, Jana Burgerová: Design of a Multi-Role AI Chatbot Framework for Supporting Project-Based Learning in Primary Education – Návrh rámca multi-rolového AI chatbota na podporu projektového učenia v primárnom vzdelávaní.	43
4. Norbert Annuš, Tibor Kmet': Overview of Artificial Intelligence Applications in the Field of Personalized Education – Mesterséges intelligencia alkalmazások lehetőségeinek áttekintése a személyre szabott oktatás területén. (invited speakers)	57
5. Krisztina Czakóová: Artificial intelligence in application development – Mesterséges intelligencia az alkalmazásfejlesztésben NP? (invited speaker)	70
6. Csilla Prantner: Innovative Design of Digital Interfaces for Learning Support – Tanulást támogató digitális felületek innovatív tervezése)	81
7. Natalia Shumeiko: How has AI made a transformation in the education sphere? (invited speaker)	86

Modern teaching methods and educational technologies
Moderné vyučovacie metódy a vzdelávacie technológie

8. Erik Túri, Győző Horváth: Use of Web Technologies in Informatics Education (invited speakers) 96
9. Andor Abonyi-Tóth, Gaál Bence: Experience-Based Informatics with micro:bits – From Wandering Kits to Hungary’s First National micro:bit Programming Competition (invited speakers).. 108
10. Péter Bernát, Gábor Törley: Methods for Solving Task Types Using Spreadsheet and Programming (invited speakers)..... 124
11. Miroslava Cimermanová: How Multimedia Content Shapes Learning Performance in E-learning: A Comparison of Results before and after its Implementation 137
12. Simona Kumpanová: Computer Graphics and its Teaching : Practical Textbook, Exercise Book and Video Tutorials..... 149
13. Milan Štrbo: Students Cheating During Online Testing 158

Programming didactics and development of algorithmic, programming and computational thinking – Didaktika programovania a rozvoj algoritmickeho, programatorskeho a počítačoveho myslenia

14. Szalayné Tahy, Győző Horváth: Specification of Extreme Value Type Tasks Specification – Szélsőérték típusfeladatok specifikációjának specifikációja (invited speakers) 172
15. Dávid Demeter, Ladislav Végh: Automatic Feedback in Programming Education : A Comprehensive Review. (invited speakers) 207
16. Noémi Bernadett Agócs: Game Development-Based Learning in an Introductory Programming Course 216
17. Péter Bernát: Didactic Advantages of Turtle Graphics in Scratch – Analysis and Case Study 227
18. Jana Fialová – Roman Horváth: Developing Computer Literacy Among Primary School Pupils 240

19. Veronika Stoffová: Visualization of Algorithms Using the Array Data Structure – Vizualizácia algoritmov používajúcich údajovú štruktúru pole 250

Preparing teachers to use educational technologies
Príprava učiteľov na používanie vzdelávacích technológií

20. Hana Hyksová: In-service training courses to increase teachers' competencies in the field of educational robotics – Doškoloľovacie kurzy na zvýšenia kompetencií učiteľov v oblasti edukačnej robotiky 262
21. Miroslav Chráska: Future Teachers and Their use of Online Technologies – Budoucí učitelé a jejich způsob využití on-line technologií 273

Tasks and problems of resent education
Problémy súčasného vzdelávania

22. Michaela Antalová, Ján Šorman, Agnieszka Kania Analytické metódy pri charakterizácii povrchových vôd vo vzdelávaní – Analytical methods in surface water characterization in educatio 277
23. Jana Jakubčinová, Michal Virág, Michaela Kóňová, Inovácie vo vzdelávaní: implementácia nových trendov z pohľadu mladej generácie – Innovations in education: implementation of new trends from the perspective of the young generation 281
24. Ladislav Rudolf, Milan Bernát, Václav, Jan Vaněk: The use of statistical methods in educational research 291

Introduction – Úvod

The proceedings book XXXVIII. DIDMATTECH 2025 consists of selected contributions from the conference with the same name which took place on 3. – 4. Dec. 2025 at Trnava University in Trnava.

The aim of conference

The aim of the scientific professional conference XXXVIIIth DIDMATTECH 2025 is to introduce the latest findings from the field of the science of materials and technologies, including educational, information and communication technologies, and enable the participants to present the results of their own scientific research and professional activities with a special focus on the didactical aspects of education.

The conference was designed mainly for teachers who teach subjects in the area of technologies, informatics, mathematics, physics, electronics, sciences and others at different schools and use modern digital technologies, ICT and artificial intelligence in education, also for doctoral, postgraduate, and for talented students.

Modern teaching methods and educational technologies

Moderné vyučovacie metódy a vzdelávacie technológie

METHODS FOR SOLVING TASK TYPES USING SPREADSHEET AND PROGRAMMING

Péter BERNÁT, Gábor TÖRLEY, HU

Abstract: One of the primary goals of the Information Technology (Computer Science) subject is to develop students' problem-solving thinking. Tasks related to data storage and processing constitute a significant and thoroughly covered topic within the subject, and they are often solvable using both spreadsheet software and programming languages. In our previous articles, we highlighted the theoretical possibilities of linking the teaching of spreadsheets and programming. In this publication, we explore the various possible solutions in spreadsheets and in programming for typical data-processing tasks and problems, as well as their conceptual and methodological connections.

Keywords: spreadsheet, programming, problem solving, algorithmic thinking, teaching methods.

1 Introduction

Within the subject of Information Technology (Computer Science), one of the most straightforward ways to develop problem-solving skills is teaching programming, which is very effective for improving algorithmic thinking and abstraction skills. In contrast, spreadsheets are often seen mainly as an application-focused area, usually connected only to mathematics. Many people think the main link between the two topics appears in macro programming.

In our earlier papers [1] [2], we compared the conceptual systems of spreadsheets and programming on a theoretical level. In this paper, using the case study method, we show the didactic connection between the two areas through solutions to specific tasks at different levels of abstraction. For each task, we first analyze the solution in spreadsheets and then the one that can be created with programming, following the practice of public education, where spreadsheets are taught earlier than programming.

Within both spreadsheets and programming, we present the solutions using specific tools: for spreadsheets, we chose Microsoft Excel, the most used system in Hungarian public education, and for programming, we used the C#

language. We chose C# partly because it can be used in the Hungarian final exam, and partly because it is the language used in the introductory programming courses of the ELTE computer science and computer science teacher training programmes. Next to most of the C# code, we also provide pseudocode, since the programming language itself is only a tool, and its specific features do not affect our basic programming thinking.

2 Literature review

According to Szalayné [3], the tables created in spreadsheet software can be seen as programs that contain data and predefined algorithms. Although students appear to work with tables, they need to understand and create “programs”, meaning solutions with functions. In this way, spreadsheets can be used to teach programming indirectly.

Bíró and Csernoch state [4] that spreadsheet software can be used as a tool for problem-solving, and their Sprego method is suitable for developing students’ computational thinking and algorithmic skills.

Many basic programming concepts have their counterparts in the terminology of spreadsheets. Kankuzi and his colleagues suggest [5] that teaching spreadsheets before introducing programming can be beneficial, because the fundamental programming concepts can be introduced indirectly through spreadsheet-related problems.

According to Warren [6], if students first use spreadsheets and then move on to a chosen programming language, they can reach more complex algorithms more quickly within the curriculum.

3 Task types

In public education, both spreadsheet and programming topics typically involve tasks related to data sets. In spreadsheets, these data sets are almost always one (or more) tables, while in teaching programming, tasks related to one-dimensional data structures (such as arrays) usually come before tasks involving multi-dimensional data structures. At the same time, in both subjects, it is common to group tasks into types based on their solution methods.

For example, let us consider a table as a data set that summarizes the oral and written scores of ten language exam candidates in a Hungarian basic-level language exam. For both parts of the exam, a minimum of 30 points out of the maximum 50 was required to pass (Table 1).

Table 1: The initial table used for the tasks

	Name	Oral	Written
01	Marie Dubois	28	38
02	Luc Moreau	9	17
03	Emily Johnson	15	25
04	Carlos García	32	38
05	Anna Schmidt	24	23
06	Li Wang	28	21
07	Camille Lefèvre	46	27
08	Oliver Smith	31	49
09	Lukas Müller	27	40
10	Wei Zhang	42	50

For this data set, the following task types and specific tasks can be given as examples (Table 2).

Table 2: Possible task types and tasks related to the initial table

Task type	Task
Sorting	Sort the data in descending order by oral exam scores!
Aggregation	What was the average score of the candidates in the oral part?
Filtering	Display only those who passed the oral part!
Conditional Aggregation	What was the average score in the written part for those who passed the oral part?
Counting	How many candidates passed the oral part?
Decision	Is there anyone who passed the oral part?
Selection / Lookup	How many points did the candidate with ID 07 get in the oral part?

In programming, the task types listed above can be associated with type algorithms or combinations of them. The Digital Culture 10 textbook [7], based on the 2022 National Curriculum (NAT), also lists type algorithms: sequence calculation (summation and averaging), decision, selection, search, counting, and finding the maximum or minimum.

In university-level teaching, in our introductory programming course, these type algorithms are called programming theorems, because their correctness can be mathematically proven. Task classes can be assigned to these type algorithms. This classification is useful because understanding the solution method for one task type allows us to solve all tasks in that class. Specific tasks can always be traced back to one of the task types. At the university level, two additional task types are added to the six mentioned above [8]: copying (function calculation) and filtering.

We will demonstrate in detail how different levels of abstraction can be used to solve the same problem in spreadsheets and programming, and what didactic connections can be found and exploited between these solutions, using the counting task type and, as a supplement, the *decision* task type.

4 Counting and its subtypes

Within each task type, different subtypes can also be defined. For example, tasks belonging to the *counting* type can be distinguished based on the type of condition (Table 3)

Table 3: Subtypes of the counting task type and example tasks related to the initial table

Condition type	Task
Simple	How many candidates passed the <i>oral</i> part?
AND condition	How many candidates passed <i>both parts</i> of the exam?
OR condition	How many candidates passed <i>at least one part</i> of the exam?
Calculated	How many candidates scored at least 70 points <i>in total</i> ?

In our study, we will focus on the first subtype within the counting task type. We will explore various ways to answer the question: “*How many candidates passed the oral exam?*” Our focus is not on the different ways of defining conditions, but on the relationship between solutions of different abstraction levels in spreadsheets and programming.

5 Level 1 solution for counting

5.1 Spreadsheets

To work with the initial table in a spreadsheet, we need to place it on a worksheet, for example starting from cell A1 (Table 4). The Excel formulas presented below will refer to the cells in this worksheet.

Table 4: Initial data in Excel

	A	B	C	D
1	ID	Name	Oral	Written
2	01	Marie-Dubois	28	38
3	02	Luc-Moreau	9	17
4	03	Emily-Johnson	15	25
5	04	Carlos-García	32	38
6	05	Anna-Schmidt	24	23
7	06	Li-Wang	28	21
8	07	Camille-Lefèvre	46	27
9	08	Oliver-Smith	31	49
10	09	Lukas-Müller	27	40
11	10	Wei-Zhang	42	50

Spreadsheet applications provide built-in functions for performing counting-type operations. In Excel, the type of condition must be considered when choosing the proper function. In the question *“How many candidates passed the oral exam?”* the condition is simple, as we want to count those who passed the oral exam. In this case, the question can be answered using the COUNTIF function (Listing 1).

=COUNTIF(C2:C11, ">=30")

Listing 1: Level 1 solution for counting in Excel

5.2 Programming

Within the field of programming, creating an algorithm must always be preceded by designing and implementing the appropriate data structure. Unlike spreadsheet applications, we do not have a complex data type that would allow us to reference every cell in a table while also expressing the logical relationship and meaning of data within rows. A matrix as a data

structure could only satisfy the first requirement. Furthermore, it has an additional limitation: by definition, all its elements must be of the same type, which prevents handling different kinds of data together (for example, names and scores).

For this reason, in programming we will use an array whose elements are records. This way, indexing of elements (essentially the rows) is preserved, and the fields of the record will express the logical relationship and meaning of the elements within a row. Note that in our introductory courses we do not introduce type algorithms (schema algorithms, programming theorems) using records, but the concepts described in this article can also be applied to simpler data structures, such as arrays of integers.

Below (Listing 2), you can see the data structure corresponding to the initial table, as well as the array declaration, where the variable *count* represents the number of candidates.

```
struct language_exam
{
    public int id;
    public string name;
    public int oral;
    public int written;
}
language_exam[] exams = new language_exam[count];
```

Listing 2: Data structure of the initial data in C#

In C#, aggregate functions, lambda calculus, and LINQ provide tools that allow us to use built-in functions to solve the task. Similar or identical tools are available in many other programming languages as well.

Using built-in functions, we can answer the question in C# as shown in Listing 3.

```
int passedOralCount = exams.Select(element =>
    element.oral).Count(result => result >= 30);
```

Listing 3: Level 1 solution for counting in C#

Using built-in functions, we can answer the question in C# as shown in Listing 3. From the data, we select the *Oral* column (*Select*) and count (*Count*) the scores that are greater than or equal to 30. The reasoning is the same as in spreadsheets: we count the appropriate elements. The main difference is that while in a spreadsheet we specify the range of data in a formula, in programming we select the appropriate record field (the “column”) by name.

6 Level 2 solution for counting

The direct solution given above can be divided into two sequential steps: first, filter the candidates who passed the oral exam, and then count only them.

6.1 Spreadsheets

An illustrative (but not automatically updating) way to perform these two steps in Excel is to use the filter option and then read the count of the selected data: by applying a “*greater than or equal to*” number filter to the *Oral* column, we can first display only the rows of candidates who passed the oral exam. Then, for example, after selecting the oral scores, we can read the count of the selected data in the status bar at the bottom of the Excel window.

The same two steps can be implemented in a way that automatically updates when the initial data changes, by using the FILTER and COUNT functions consecutively. First, the FILTER function can return the oral scores that are 30 or higher, and then the number of elements in the resulting array can be determined using the COUNT function (Listing 4).

```
=COUNT(FILTER(D2:D11,D2:D11>=30))
```

Listing 4: Level 2 solution for counting in Excel

6.2 Programming

In C#, this level of solution is not typical because a built-in function (Listing 3) exists for counting. However, some languages (such as JavaScript) do not provide such a high-level function that returns the number of elements in a sequence meeting a certain condition.

The second-level solution can also be created using built-in functions in C# (Listing 5).

```
int passedOralCount = exams.Where(element =>  
    element.oral>=30).Select(element =>  
    element.oral).ToArray().Length;
```

Listing 5: Level 2 solution for counting in C#

7 Level 3 solution for counting

7.1 Spreadsheets

A lower-level solution can be created by implementing the FILTER function using simpler functions and a helper column. For the question we are

studying, the following approach can be used. First, fill a helper column to the right of the main table with an IF function so that it contains only the oral scores of candidates who passed the oral exam, leaving the other cells empty (""). Second, count the numbers in this column (Table 5). (Alternatively, we could write 1s and 0s in the helper column for the two cases and then use the SUM function instead of COUNT to determine the number of entries.)

Table 5: Level 3 solution for counting in Excel

	C	D	E	F	
1	Oral	Written			
2	28	38			=IF(C2>=30,C2,"")
3	9	17			=IF(C3>=30,C3,"")
4	15	25			=IF(C4>=30,C4,"")
5	32	38		32	=IF(C5>=30,C5,"")
6	24	23			=IF(C6>=30,C6,"")
7	28	21			=IF(C7>=30,C7,"")
8	46	27		46	=IF(C8>=30,C8,"")
9	31	49		31	=IF(C9>=30,C9,"")
10	27	40			=IF(C10>=30,C10,"")
11	42	50		42	=IF(C11>=30,C11,"")
12					
13				4	=COUNT(F2:F11)

7.2 Programming

The previous reasoning does not appear with static arrays, as it would lead to the selection algorithm, which “implicitly” includes counting (at the end of the algorithm, the number of selected elements appears in an auxiliary variable), making it unnecessary to store the selected elements separately. However, if we implement the selection algorithm using a dynamic array type, the result of the algorithm does not explicitly reveal how many elements were selected: we need to query the size of the resulting dynamic array (Listing 6).

This approach is mainly relevant for languages that do not support static arrays (such as Python).

Algorithm

```

Variable
  i: Integer
passedStudents := []
For i:=1 to count do
  If exams[i].oral >= 30
  then
    insertAtEnd(
      passedStudents,
      exams[i].oral)
End For
passedOralCount :=
passedStudents.size
    
```

Code

```

int passedOral = 0;
List<int> passedOralResults =
new List<int>();
for (int i = 1; i <= count; i++)
{
  if (exams[i - 1].oral >= 30)
    {passedOralResults.Add
(exams[i - 1].oral);}
}
passedOralCount =
passedOralResults.Count;
    
```

Listing 6: Algorithm and C# code of the level 3 solution for counting

8 Level 4 solution for counting

The idea for a more efficient algorithm comes from the method of mental calculation. If the table were only available in printed form and we had to determine the number of candidates who passed the oral exam in our heads, we could follow this algorithm: go through the *Oral* column one by one from top to bottom, and each time a value is at least 30, add one to a running count (which starts at 0). After processing the last cell of the column, the running total will be the result. Note that this solution, unlike the one described in the previous section, does not create a separate data structure containing the oral scores that are 30 or higher.

8.1 Spreadsheets

The solution described above can be implemented in Excel as follows (Table 6). The result is shown in cell F12.

Table 6: Level 4 solution for counting in Excel

	C	D	E	F
1				Count
2	Oral	Written		0
3	28	38		0 =IF(C3>=30, F2+1, F2)
4	9	17		0 =IF(C4>=30, F3+1, F3)
5	15	25		0 =IF(C5>=30, F4+1, F4)
6	32	38		1 =IF(C6>=30, F5+1, F5)
7	24	23		1 =IF(C7>=30, F6+1, F6)
8	28	21		1 =IF(C8>=30, F7+1, F7)

9	46	27		2	=IF(C9>=30,F8+1,F8)
10	31	49		3	=IF(C10>=30,F9+1,F9)
11	27	40		3	=IF(C11>=30,F10+1,F10)
12	42	50		4	=IF(C12>=30,F11+1,F11)

8.2 Programming

This reasoning leads us to the counting algorithm. We examine each element in turn, and if the current element meets the condition, we increment a variable (initialized to 0) called *passedOralCount* by one (Listing 7).

Algorithm	Code
<pre> Variable i: Integer passedOralCount := 0 For i:=1 to count do If exams[i].oral >= 30 then passedOralCount := passedOralCount + 1 End For </pre>	<pre> int passedOralCount = 0; for (int i = 1; i <= count; i++) { if (exams[i - 1].oral>=30) { passedOralCount++; } } </pre>

Listing 7: Algorithm and C# code of the level 4 solution for counting

In spreadsheets, the role of the IF function (Table 6) corresponds to the conditional control structure in the algorithm. If the current array element meets the condition, we add 1 to the current count; otherwise, we do nothing.

9 Level 4 solution for decision

After the detailed discussion of the counting task type, we will briefly address the decision task type. For the specific initial table and the previously examined question “*How many passed the oral part?*”, the related decision-type question is: “*Is there anyone who passed the oral part?*” This is a yes-or-no question: the answer only needs to indicate whether someone passed, not how many.

We discuss this task type for two reasons. First, Excel does not provide a single function to solve it directly; instead, one must count the entries and then check whether the result is at least 1. Second, in programming, the same two-step solution is possible, but it is not the most efficient approach and therefore does not represent the task type’s type algorithm.

For this task type, we want to highlight the insight observable at the fourth level, so we will now present only the level-4 solution in detail for both spreadsheets and programming.

9.1 Spreadsheets

The basic idea of the level-4 solution for the decision task type can also be to maintain a running result with an appropriate initial value, updating it as needed. We can realize that the running result should indicate whether there has already been a value meeting the condition up to the current value.

For the specific question *“Is there anyone who passed the oral part?”*, the running result for any candidate should be TRUE if either one of the previous candidates has already passed the oral part, or the current candidate passed it. This solution can be implemented as shown (Table 7). The result can be read in cell F12, although it already appears in cell F6.

Table 7: Level 4 solution for decision in Excel

	C	D	E	F
1				Exists
2	Oral	Written		FALSE
3	28	38		FALSE =OR(F2, C3>=30)
4	9	17		FALSE =OR(F3, C4>=30)
5	15	25		FALSE =OR(F4, C5>=30)
6	32	38		TRUE =OR(F5, C6>=30)
7	24	23		TRUE =OR(F6, C7>=30)
8	28	21		TRUE =OR(F7, C8>=30)
9	46	27		TRUE =OR(F8, C9>=30)
10	31	49		TRUE =OR(F9, C10>=30)
11	27	40		TRUE =OR(F10, C11>=30)
12	42	50		TRUE =OR(F11, C12>=30)

9.2 Programming

The novelty in the fourth-level solution for the decision task type – and this can be observed by comparing it to the spreadsheet solution – is that once the running result changes from FALSE to TRUE, it is unnecessary to process further candidates. The spreadsheet application cannot “think” this way because its functions always process all data, whereas in programming we can terminate processing when appropriate.

When teaching programming fundamentals, we clearly distinguish between task types that use a counting loop and those that use a pre-test loop, and based on the efficiency considerations above, we may conclude that a new

loop type is needed. Nevertheless, many beginner programmers find it more natural to use a loop that processes *all* elements and sets a Boolean variable (initially FALSE) to TRUE when the condition is met for the current element. The spreadsheet example, however, may encourage us to process elements only until the desired condition is satisfied. The pre-test loop and the decision algorithm work exactly this way (Listing 8).

Algorithm

```
Variable
  i: Integer
i := 1
While i <= count and
exams[i].oral < 30
  i := i + 1
End While
exists := i <= count
```

Code

```
int i = 1;
bool exists;
while (i <= count &&
exams[i-1].oral < 30)
{
  i++;
}
exists = i <= count;
```

Listing 8: Algorithm and C# code of the level 4 solution for decision

10 Conclusion

In this paper, we primarily aimed to show that in teaching both spreadsheets and programming, it is worthwhile to examine and develop solutions to task types at different levels of abstraction. On the one hand, this is because solutions at different levels often show significant similarities (and sometimes instructive differences) across the two areas, allowing students to transfer knowledge from one area to the other. On the other hand, it demonstrates that typically multiple levels of solution exist for a problem, often with connections between them. Moreover, higher-level solutions can help students understand the essence of a problem, while lower-level solutions can guide them toward more efficient approaches.

We also wanted to show that spreadsheets are not merely an application tool; even before learning programming, students encounter fundamental algorithmic concepts such as conditions, iteration, counting, and selection.

In public education, students usually become familiar with spreadsheets before type algorithms (programming theorems). This provides an opportunity to build teaching programming on existing spreadsheet knowledge and to use spreadsheets consciously during the introduction of programming.

In this paper, we examined the counting and decision task types, but our research plans include applying a similar method to other standard task types as well.

Our final message is that spreadsheets and programming are not two isolated areas but mutually reinforcing tools. Exploring solution paths at different levels of abstraction ultimately enables students to develop comprehensive, flexible, and adaptable problem-solving skills.

References

1. TÖRLEY, G. – ZSAKÓ, L. – BERNÁT, P.: Didactic Connection between Spreadsheet and Teaching Programming. In *Athens Journal of Technology and Engineering*, 9. 2., 2022, pp. 77-94.
2. TÖRLEY, G. – BERNÁT, P.: Spreadsheet As An Algorithm Visualization Tool. In *XXIV. DidMatTech*, Budapest, ELTE Faculty of Informatics, 2021, pp. 15-27.
3. SZALAYNÉ TAHY, ZS.: How To Teach Programming Indirectly – Using Spreadsheet Application. In *Acta Didactica Napocensia*, 9. 1., 2016, pp. 15-22.
4. BÍRÓ, P. – CSERNOCH, M.: Algoritmusok és/vagy táblázatkezelés? In *VII. Oktatásinformatikai Konferencia Tanulmánykötet*, 2015, Budapest, ELTE Pedagógiai és Pszichológiai Kar, pp. 97-111.
5. KANKUZI, B. – ISONG, B. – LETLONKANE, L.: Using the spreadsheet paradigm to introduce fundamental concepts of programming to novices. In *Proceedings of SACLA'17*, Potchefstroom, North-West University, 2017, pp. 39-45.
6. WARREN, P.: Learning to program: spreadsheets, scripting and HCI. In *Proceedings of the Sixth Australasian Conference on Computing Education*, vol. 30, Darlinghurst, Australian Computer Society, Inc., 2004, pp. 327-333.
7. PINTÉR, G. (ed.): *Digitális kultúra 10. tankönyv*. Oktatási Hivatal, 2020, ISBN 978-615-6256-39-3 [on-line] https://www.tankonyvkatalogus.hu/storage/pdf/OH-DIG10TA_teljes.pdf
8. SZLÁVI, P. – TÖRLEY, G. – ZSAKÓ, L.: Programming theorems have the same origin. In *Central-European Journal of New Technologies in Research, Education and Practice*, 2019, 1. 1., pp. 1-12.

Contact address

Dr. Péter Bernát, PhD., Dr. Gábor Törley, PhD.
Eötvös Loránd University, Faculty of Informatics,
Department of Media and Educational Technologies
Pázmány P. stny 1/C, H-1117 Budapest,
bernatp@inf.elte.hu, gabor.torley@inf.elte.hu

Editors: Prof. Ing. Veronika Stoffová, CSc.
PaedDr. Krisztina Czakóová, PhD.
Mgr. Ing. Roman Horváth, PhD.

Graphic editors:

Title: XXXVIIIth DIDMATTECH 2025
**New methods and technologies in
education, research and practice
Proceedings**

Pages: 295 pages

Preparing to print: Authors of proceedings &
Prof. Ing. Veronika Stoffová, CSc.
PaedDr. Krisztina Czakóová, PhD.
Mgr. Ing. Roman Horváth, PhD.

Print:

First edition

ISBN: 978-80-568-0796-5

EAN: 9788056807965