Algorithmic thinking in paradigms of programming

Stepan Hubalovsky and Ondrej Korinek

Abstract**—**Everyone faces the problems in everyday activities. They can be of various kinds - personal, business or other. To solve the problem, it is necessary to find a procedure, process that will solved it. It is necessary to establish the algorithm. Algorithms can be found not only in everyday routine activities, e.g. during crossing the street through the transition, cooking food etc., but also in subjects of programming. A number of courses of programming at different schools starts the teaching with algorithm development. The algorithm development is primarily related with structured paradigm of programming. On the other hand modern and most widely used is object-oriented paradigm in programming

Teaching methodology and election of paradigm of programming always depends on the particular school and taught subjects. The algorithm development is, for example bases of subjects like graph theory. The algorithm development has its place in teaching of programming. The proposal of procedures of problem solution is closely related to the way of thinking that beginning programmers are used.

The paper describes algorithmic thinking and analyzes the results of two teaching methodologies related to algorithm development - structured and object oriented paradigm versus object oriented paradigm with regard to algorithmic thinking of students of the Faculty of Science, University of Hradec Kralove.

Keywords**—**Algorithm development, algorithm thinking, structured paradigm of programming, object oriented paradigm of programming.

I. Introduction

THE development of new programming languages are often associated with new paradigms in programming. The languages of lower level, was replaced by the high-level languages [1]. The supporters and opponents of two most widely used paradigms - structured programming and object-oriented programming discussed the advantages and disadvantages of both paradigms. Recently, the most widely used is object-oriented paradigm, which is usually required by companies in the labor market. A candidate who can use the object libraries and creates the object program with under the rules of design patterns, interfaces and inheritance, has great advantage with comparison of candidate who cannot used it. Creation of extensive application using structured programming is not possible. In spite of this fact the object-oriented programming is taught in number of schools in later phase the programming or in optional subjects. Opponents of such a methodology (e.g. [2]), often point the fact that the student completely doesn't understand the object programming and often programmed by the previous ingrained habits. Despite the entirely legitimate objections against the structured oriented methodologies of teaching of programming, the introduction to structured oriented programming has their place. Other commonly used methods of teaching, is algorithm development and structured programming. At some schools algorithm development continues by object graph theory [3]. Structured programming is used e.g. in programming of robots in Lego Mindstorms.

Stepan Hubalovsky is assoc. prof. and supervisor of Ondrej Korinek. He works at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, stepan.hubalovsky@uhk.cz.

Ondrej Korinek is Ph.D. student at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, ondrej.korinek@uhk.cz.

II. Algorithm Development and Programming

Algorithm can be represents in several ways - in the form of flowcharts, pseudocodes or structure-grams. It always depends on the teacher, which type of algorithm representation prefer [4]. Algorithm development is the basis of programming [5].

The first aspect that influences learning of algorithms and programming is influenced by the form of performed teaching:

- structured form, that teaching is divided into learning algorithms, structured programming and object-oriented programming in the end;
- object oriented form, i.e. from the beginning of the instruction focuses on object-oriented programming, with the principles of the algorithms are part of this instruction.

The paper analyzes the results of two teaching methodologies related to algorithm development based on structured programming with regard to algorithmic thinking, so let's described basic paradigm of structured programming first.

Structured programming is programming based on the structure of the program, which comes strictly from the algorithm flowchart. From the system approach point of view the algorithm as well as structured program (written in any structured language - Pascal, C++, VB Script) can be understood as system, because they have properties of the system – algorithm interacts with its environment through inputs and outputs, consists from elements that are affected by interactions. Another division algorithm to subsystems is possible, from a practical perspective, however unreasonable.

In this context it is necessary to mention what types of exercises are used for training the algorithm development and structured programming. The exercises reflect two facts. Firstly, in the past, early in the courses of programming (mainly structured programming - Pascal, Basic, etc.) has been teaching of programming realized by teachers who also taught mathematics, or had to mathematics very close. Second, math problems are basically the simplest tasks, can be clearly described, defined and then developed by algorithm and rewritten to the program structure. That, however, seems at first glance a logical and simple, brings disadvantages. Algorithm development and structured programming explained by the mathematical tasks usually focus on rewriting the mathematical equations and formulas to the algorithms regardless of their complex systems integration with the exercises from real life. Used tasks are often artificial and divorced from reality. System and multidisciplinary approach is missing. Students, who do not have sufficient mathematical skills, do not understand the task and it can result in resistance to the algorithm development and subsequently to programming.

Despite the different representation of algorithm and different paradigm of programming students should improve algorithmic thinking.

III. Algorithm Thinking

When designing the algorithm the various terms are used and combine. The terms are related to previous practical experience and theoretical knowledge of programmer. Algorithms that programmers solve, in most cases are also problematic. The algorithmic thinking is important for the proper design and construction of the algorithm, in which it is necessary to take into account the time and memory consumption [9]. The algorithm thinking is also used when analyzing e.g. the best sorting algorithm of the sequence of numbers or to verify that designed procedure satisfies all the properties of the algorithm. Algorithm thinking can be related to the basic terms that are used in algorithms development.

A. The Forms of Learning in Algorithm Thinking

Among the forms of learning that are related to algorithmic thinking, will include: deduction, induction, sorting, comparison, analysis and synthesis [10].

Deduction at algorithmic thinking used e.g. in the design of algorithms. The algorithm must meet certain rules that are universal and that has to be applied to build required algorithm.

Induction at the algorithmic thinking is used in reverse case than in the previous proposal, e.g. if in the design of a required algorithm is needed to check the general steps to prove the procedure is algorithm.

Sorting is in algorithmic thinking used for sorting algorithms. The sorting algorithm are the basis for teaching of algorithms. It is always necessary, with the given values in the sequence to determine, which sorting algorithm is most appropriate for a given sequence.

Comparing in algorithmic thinking can be understood as the most important form of algorithm learning. When teaching the algorithm development, the basis is to propose the most efficient algorithm that solves the problem. In proposal of the algorithm, the student has to use comparison thinking skills to select the most efficient algorithm from several proposed algorithms that solves the problem.

Analysis and *synthesis* are one of the most important intellectual operations that are related together [3]. At the beginning of the design of the algorithm the analysis of use of algorithmic structures and elements in the algorithm should be provided.

IV. Research of Methods of Teaching in the Subject of Programming

A. Methodology of the Research

Students at the Faculty of Science Univerzity of Hradec Králové in the study field Informatics in Education meet with programming in the first semester of the course Algorithms and Data Structures (hereinafter ALGDS). The course deals with basic algorithmic structures, one-dimensional array, matrixes and algorithms for sorting. The course of ALGDS is followed by three courses of programming in from the second up to fourth semesters. Programing language is C#.

The research investigation was carried out in the course of programming in the academic year 2013/2014. The main goal of the research was determined the comparison of two methods of teaching of programming - object-oriented programming and structured versus object-oriented programming with respect to algorithmic thinking. Students were randomly divided evenly into two groups according to the results in course ALGDS.

One group of students followed the algorithm development by structured programming in C # programming language with functions (methods) and based on algorithmic structures. The students designed structured C# programs based on similar algorithms, the already developed in course of ALGDS. The programs included conditions, loops, arrays, matrices. The structured programming was then followed by object-oriented programming, where the basic concepts of OOP were discussed.

The second group of students began immediately after the algorithm development (after the course ALGDS) with object oriented programming (without structured programming). In this group the concept of object oriented programming was more practiced. The concept of structured programming was omitted.

Both groups of students passed a midterm exam test with similar tasks, which consisted of theoretical and practical part. Practical (programming) part was divided into object and algorithmic part. To successfully pass the test, students had to reach in every part at least 60% of correct answer.

B. Credit Test

The credit test consists form some different tasks. Students have to correctly designed class first. They initialize one dimensional array (sequence) by constructor. Algorithm constructions for one dimensional array create classes in methods. Methods for correct design of algorithm are focused to following areas:

- input data to the sequence;
- output data from the sequence;
- calculations and search of value in the sequence;
- shifts the values in the sequence,
- inserting / removing values in the sequence;
- work with multiple sequences.

Students has to fulfill one task from each area and create algorithm. Class definition, constructors and methods are separated from algorithmic structures in evaluation of the task.

Sample of credit test:

Create a new project in C # console application In Visual Studio based and fulfill following assignment.

Create class Sequence for sequences operation (one-dimensional array) with the following components:

Basic algorithms:

- *Constructor* - creates private data item of one-dimensional array type of integers of a given size.
- *N* - read-only property specifying the length of the sequence.
- *Fill* - filled array by numerical series in two different ways (overloads):
  - initial value will be set by input parameter (range will be from *X* to *N* + *X*).
  - input parameter is missing and a series will start from 1 (up to *N*).
- *WriteRow* - writes sequences to row of console (values are separated by commas).
- *Member* - returns values of sequence member in position specified by input parameter.

Algorithm for Calculations and Search:

- *Number* - determines the number of members whose value is equal to the value specified as input parameter and returns this number as output value

Algorithm for shift of values:

- *CycleShift* - cyclically moves to right the members of the sequence.

Algorithm for inserting the values:

- *Insert* - insert into the sequence the member whose value and position will be determined by input parameter.

Algorithm for work with multiple arrays

- *Division* – selects all the values of sequence members that are divisible by the value specified as input parameter and returns a value of sequence type as output.

In the main part of the program (method Main Class Program), create instance of the class Sequence and properly use all implemented methods.

C. The Result of the Research

The research investigated the influence of the different concepts on increase of algorithmic thinking.

The first credit test was the same for both groups of students. The test examines practical skills operate with one-dimensional array - calculations and searches, shifts the values in the sequence, inserting / removing values and work with multiple fields.

In the first group of the students (course of ALGDS followed by course of structured and object-oriented programming – *algorithmic group of student*) consist of 9 students participated in the test.

In the second group of the students (course of only OOP – *object oriented group of students*) consist of 8 students participated in the test.

Result of the algorithmic test

To determine whether the median of the result of student achieved in algorithmic part is the same for the first and second groups of students the nonparametric Mann-Whitney test was used.

Calculated P- value is P = 0.030384

with significance level α = 0.05,

so we can reject the null hypothesis that the median of students results of the algorithmic part between the groups is the same. Between groups is statistically significant difference. Box plots diagram of both groups of students is in figure 1.

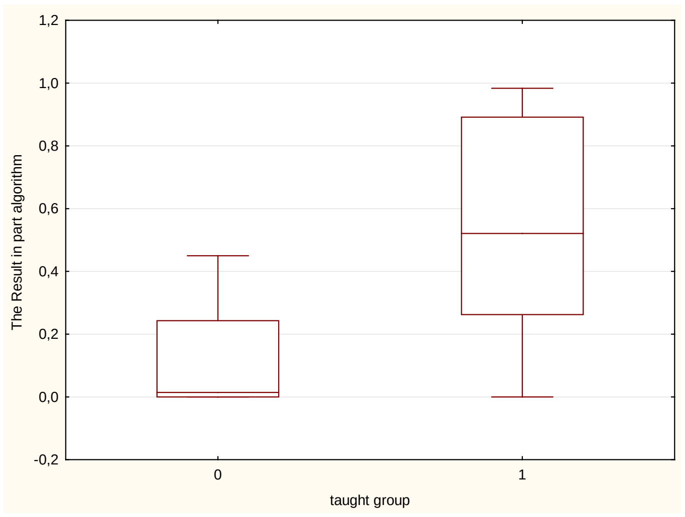


Fig. 1 Box plot diagram comparing result of algorithmic group of students with object oriented group of students in algorithmic test.

The results show that the first algorithmic group of students reached far worse results than the second object oriented group of students.

All student failed in the first algorithmic group.

Half of students (50%) succeeded in the second object oriented group of student.

Interesting results can be also reached from analysis of the code of programs (will be published later).

Algorithmic tasks of the test can be divided into two parts.

The *first part* contains tasks testing following terms: definition, input and output of the sequence in the form of one-dimensional array. These tasks was more trained in the algorithmic group of students.

The *second part* contains algorithmic construction for one-dimensional array: e.g. calculations and search, shifts of the values in the sequence etc. These task was practiced in both the courses – course ALGDS and programming.

Expected result should be as follows:

- first algorithmic group of students should have better results in the first part of the test
- the second part of the test should have the similar results

Result of the first part of algorithmic test

To determine whether the median of the results of the first part of the test (algorithmic part) (variable definition, input and output to the sequence) is the same for the both groups of students. It was again calculated by the nonparametric Mann-Whitney test.

Calculated P-value is P = 0.0237

with significance level α = 0.05,

so we can reject the null hypothesis that the median of the results of the first algorithmic part of test algorithmic between both groups of student is the same. Between groups is statistically significant difference. Box plots chart of the two groups of students is shown on Figure 2.

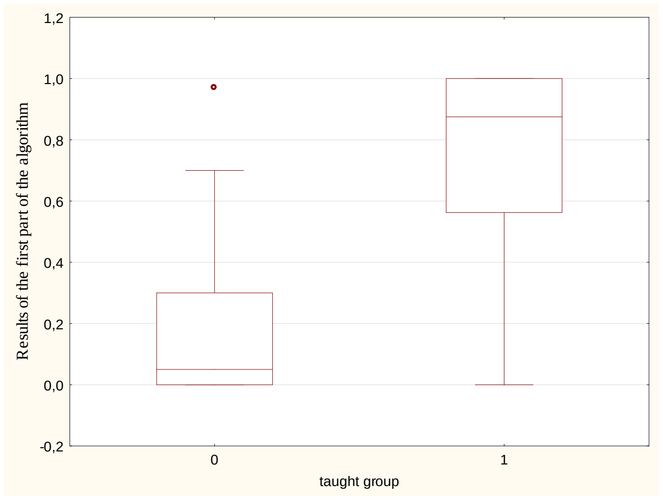


Fig. 2 Box plot diagram comparing result of algorithmic group of students with object oriented group of students in the **first** part of algorithmic test.

The graph shows that the value of median of students from the second OOP group of students is greater than the maximum value of students from the first algorithmic group of students, excluding outliers.

Result of the second part of algorithmic test

To determine whether the median of the results of student of the second part of the test (concerning the sequences) is the same for the first and second group of students was again used the nonparametric Mann-Whitney test.

Calculated P-value is P = 0.075

with significance level α = 0.05,

so we cannot reject the null hypothesis that the median of results of the students from the second algorithm part of the test is the same. Among groups there is not statistically significant difference.

Box plots graph of both groups of students is shown on Figure 3. From the graph it is clear that the second group of students gained better results than the first group.

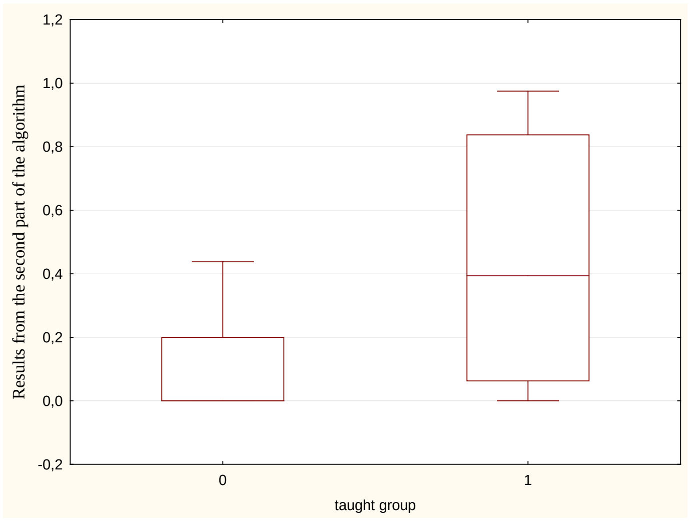


Fig. 3 Box plot diagram comparing result of algorithmic group of students with object oriented group of students in the **second** part of algorithmic test.

D. Sample of Result of Tests

Students do not have problems with:

- algorithms for filling the sequence;
- output the value from sequence;
- algorithm for search of value in the sequence;
- algorithm for shifts value in the sequence;
- using of cycles;
- storing and writing the values to other variables;

Students have problems with:

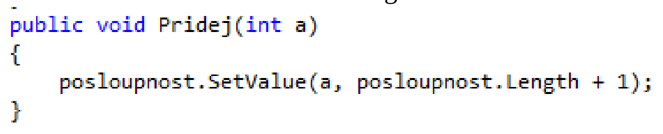
- algorithm for inserting members to the sequence;
- algorithm for removing members from the sequence.

The problem is with changing the size of the field with determination or setting the size of the resulting sequence.

Example of task 1:

Create method that adds at the end of the sequence member whose value will be determined by input parameter.

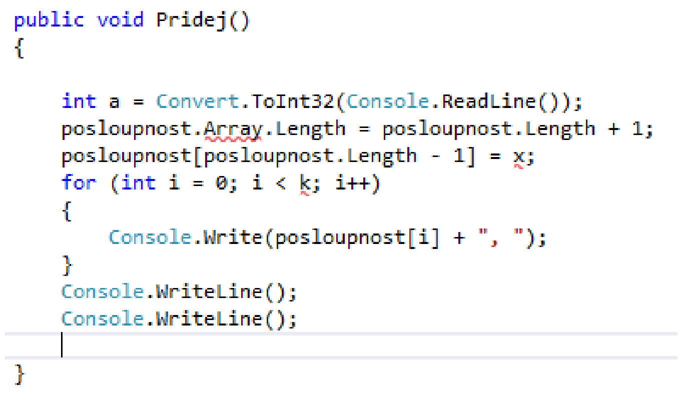
The correct solution is shown in figure 4.



```
public void Pridej(int a)
{
    posloupnost.SetValue(a, posloupnost.Length + 1);
}
```

Fig. 4 Method – add member at the end of the sequence – correct solution.

Wrong student's solution is shown in figure 5 (wrong using of field sequence wrong assignment to the field).



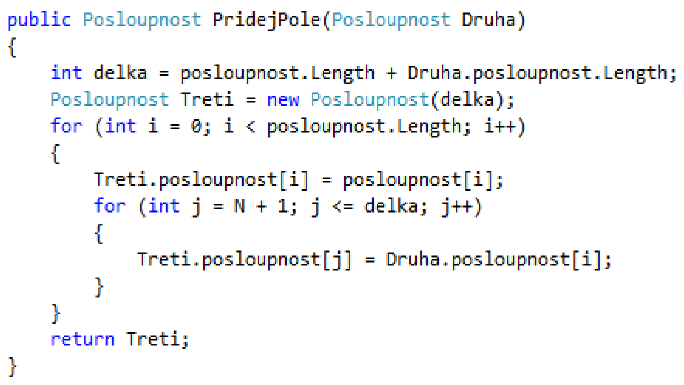
```
public void Pridej()
{

    int a = Convert.ToInt32(Console.ReadLine());
    posloupnost.Array.Length = posloupnost.Length + 1;
    posloupnost[posloupnost.Length - 1] = x;
    for (int i = 0; i < k; i++)
    {
        Console.Write(posloupnost[i] + ", ");
    }
    Console.WriteLine();
    Console.WriteLine();

}
```

Fig. 5 Method – add member at the end of the sequence – wrong solution.

Students has also problems with design of algorithm development working with multiple sequences. They cannot verify the possibility of merge of two sequences and connect more sequences to different sequence.

Example of task 2:

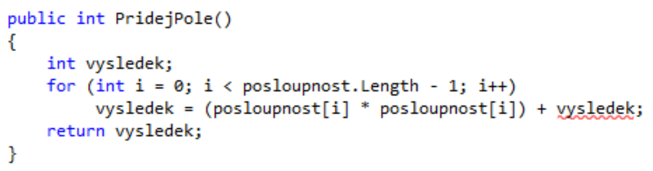
Create method that merge the sequence at the end of the second one. The second sequence is specified as input parameter to the new third sequence. The result is returned as output value of type Sequence. The correct solution is shown in figure 6.



```
public Posloupnost PridejPole(Posloupnost Druha)
{
    int delka = posloupnost.Length + Druha.posloupnost.Length;
    Posloupnost Treti = new Posloupnost(delka);
    for (int i = 0; i < posloupnost.Length; i++)
    {
        Treti.posloupnost[i] = posloupnost[i];
        for (int j = N + 1; j <= delka; j++)
        {
            Treti.posloupnost[j] = Druha.posloupnost[i];
        }
    }
    return Treti;
}
```

Fig. 6 Method – merge the sequence at the end of the second one – correct solution.

Example of wrong student solution is shown in Figure 7.



```
public int PridejPole()
{
    int vysledek;
    for (int i = 0; i < posloupnost.Length - 1; i++)
        vysledek = (posloupnost[i] * posloupnost[i]) + vysledek;
    return vysledek;
}
```

Fig. 7 Method – merge the sequence at the end of the second one – wrong student's solution.

V. Conclusion

Paper describes algorithmic thinking and compared the two methodologies of teaching of programming in relation to the algorithmic and object oriented thinking.

Based on result of our research it is clear, that no student from the first group (first structured programming than object oriented programming) succeeded in the algorithmic part of the test.

On the other hand 50 % of students from the second group (only object oriented programming) succeeded the same test.

Detailed analysis of the results of two algorithmic parts test discovered that the results of the first group was far worse despite the fact that learning in the first group was more focused on algorithm development than in the second group.

The causes of failure may be several. One factor could be underestimation of the credit preliminary test. Another factor could be in the teacher's approach, because each group was taught by different teacher. To eliminate this factor, we will provide in this academic year the same research with the same teacher for both groups.

The results provide feedback based on which the learning of algorithm and programming will be modify.

Acknowledgment

This research has been supported by: Specific research project of University of Hradec Kralove, Faculty of Education in 2015 and Specific research project of University of Hradec Kralove, Faculty of Science in 2015.

References

[1] E. D. Knuth, L. T. Pardo, "*Early development of programming languages*". Encyclopedia of Computer Science and Technology, Marcel Dekker, Vol **7**, pp. 419–493.

[2] R. Pecinovsky, "Jak efektivně učit OOP (How to teach OOP effectively)", in *Proceedings of the conference Software Development 2005*, Ostrava, Technical University of Ostrava, 2005, pp. 174 – 182.

[3] E. Milkova, "Graph theory and algorithms: various approaches to utilization of virtual learning environment", in *Proc. 14th International Conference on Interactive Collaborative Learning (ICL2011) - 11th International Conference Virtual University (VU'11)*, Piešťany, 2011, pp. 637 – 642.

[4] E. Milkova, A. Sevcikova, F. Samek, "Výuka algoritmizace – několik postřehů, rad a doporučení", in *Proc. Poškole 2004 - Sborník Národní konference o počítačích ve škole,* Poskole, Mezinárodní organizační výbor, 2004, pp. 138-142.

[5] S. Hubalovsky, "Research of Methods of a Multidisciplinary Approach in the Teaching of Algorithm Development and Programming", in *Proc.*

DIVAI 2012 – 9th International Scientific Conference on Distance Learning in Applied Informatics, 2012. pp. 147 – 156.

[6] J. A. Sajaniemi, Chenglie, “Teaching Programming: Going beyond “Objects First””, 2006 in *Proc. Psychology of Programming*, [online]. Available: http://www.ppig.org/papers/18th-sajaniemi.pdf

[7] M. Ricken, *Assignments for an Objects-First Introductory Computer Science Curriculum*, 2005, [online]. Available: http://www.cs.rice.edu/~javaplt/papers/tr200504.pdf

[8] J. Bennedsen, C. Schulte, “What does objects-first mean?: An international study of teachers' perceptions of objects-first”, in *Proc. of the Seventh Baltic Sea Conference on Computing Education Research*, Australian Computer Society, Inc., vol. 88, pp. 21-29, 2007. Available: http://crpit.com/confpapers/CRPITV88Bennedsen.pdf

[9] E. Milková, “Multimedia Tools for the Development of Algorithmic Thinking”. *Recent Patents on Computer Science*, 2011, Vol. 4, No. 2, pp. 98- 107.

[10] D. L. Schacter, D. T. Gilbert, D. M. Wegner, “*Psychology*”, Bedford, Worth Publishers, 2011.

Stepan Hubalovsky was born in Trutnov, Czech Republic in 1970, he obtained master degree in education of mathematics, physics and computer science in 1995, Ph.D. degree in theory of education in physics in 1998 both in Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic and assoc. prof. degree in system engineering and informatics in 2012 in University oh Hradec Kralove, Czech Republic.
He worked 5 years as master of mathematics, physics and computer science on several secondary schools. He works as associate professor on University of Hradec Kralove from 2006. He interested in algorithm development, programming, system approach, computer simulation and modelling.
Assoc. prof. RNDr. Stepan Hubalovsky, Ph.D. is member of Union of Czech Mathematicians and Physicist.

Ondrej Korinek was born in 1985 in Horice, Czech Republic. He graduated from the University of Hradec Kralove, Czech Republic in 2010, where he studied Education Mathematics and Computer Science for Secondary Schools.
Since 2012, he has continued his studies of Information and Communication Technology in Education at postgraduate level. Since 2010 he has been working as an ICT teacher at VOS and SPS in Jicin, Czech Republic. He is interested in algorithms, programming methodology, database systems and modern technology.