Influence of the Encryption Algorithms to Development of Algorithmic Thinking in Education of Programming

S. Hubalovsky and P. Hanzalova

***Abstract*—**The paper highlights the alternative learning of algorithm development and programming using cryptographic algorithms. The first section describes the different methods of teaching of programming. We focused primarily on those methods, which are used in our research. The next section described relationship between algorithmic thinking and cryptology. The set of examples demonstrated using of cryptology in algorithm development is part of the paper too. The last section shows the methodology and results of the pedagogical research, which takes place at the University of Hradec Kralove within specific research project, particularly we focus to the determination of the research group, methods, justification for the choice of programming language and existing research results.

***Keywords*—**Didactic of informatics, programming, algorithm development, algorithm thinking, cryptology.

I. Introduction

The interest of student on subjects focused on algorithm development and programming is no longer such as it was few years ago. At the same time, however, we can say that these subjects are taught beyond the technical or scientific subjects. Our research takes place at the Philosophical Faculty, University of Hradec Kralove in the humanities-oriented group of students. They have learning of algorithm development and programming as compulsory subject in its curriculum. It is study program computer-aided teaching archiving.

Based on previous experience within this subject, we noticed that content of the subject is for the humanities-oriented student quite difficult. This is mainly due to the high content of mathematics in algorithm tasks. Using of numeric exercises is the most common approach of learning of algorithm development and programming.

We therefore suggested and want to verify a new method that combines various existing approaches to teaching of algorithm development and programming. The main goal is to motivate students to mobilize, guide them to solve various examples. The new methodology emphasizes on clarity. At the same time we try to choose the least examples from numerical mathematics - most examples is built on cryptographic algorithms. These examples are easy to understand, can be clearly demonstrate and student can know only basic mathematical knowledge. At the same time there are a lot of such examples, we can demonstrate the basic algorithmic instructions, or even different algorithmic construction and functions.

Stepan Hubalovsky is assoc. prof. and supervisor of Pavla Hanzalova. 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.

Pavla Hanzalova is Ph.D. student at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, pavla.hanzalova@uhk.cz.

II. Approaches in Learning of Programming

A. Definition of the Approaches in Learning of Programming

There are different approaches in learning of beginners in programming. R. Pecinovsky [1] describes the following teaching programming techniques:

- *Hardware-first* = student has to first know the design of computers (hardware, machine code, and subsequently higher programming languages).
- *Algorithms-first* = student first uses a general representation of the algorithm (flowcharts, pseudo code) and subsequently writes program in the syntax of the language
- *Imperative-first* = student first learns classical structured programming instructions, and then passes the object oriented programming.
- *Functional-first* = student first learns function, which is usually not used in practice, so it discourages most of students
- *Objects-first* = student becomes from beginning familiar with object-oriented programming straight (there may student be information overloaded)
- *Breadth-first* = student has to understand first computer science in general; he has an overall view, which subsequently is applied to programming.

Most often used approach in learning of object-oriented programming is Object-first - see e.g. [2], [3], [4]. The second most common approach is Algorithms-first, often followed by learning of structured programming (see e.g. [5]).

ISBN: 978-1-61804-322-1

B. Structured Programming

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.

C. Object Oriented Programming

In contrast, the basic paradigm of Object oriented programming (OOP) is to model on the computer the real-world situation. The OOP applications are developed based on already created components. The basic terms of OOP are object, event abstraction, encapsulation, inheritance and polymorphism. From a system approach point of view the objects can be understood as open sub-system of whole application. Every object - the subsystem is a complete system - consisting of elements (a list of properties, event handlers), communicates with its environment through inputs (events, parameters) and outputs (methods and parameters).

D. Methods of Learning

The first method which respects the didactic principles is **constructivism**. The founder of constructivism is considered J. Piaget. The constructivist approach is based on cognitive methods and significantly is used individualization of teaching [6].

In constructivist teaching student itself creates new knowledge based on their previous knowledge, together with the information that can acquire during cognitive processes. The main feature of constructivism is the concept of learning as active, deliberate and social process of constructing meanings from submitted information and induced experience [7].

Constructivist method of teaching of programming is described by L. Salanci [8] or M. Ben-Ari [9]. When using constructivist teaching the student are able to solve various tasks, although they had never met them. The course consists of graded tasks and leading questions. Great emphasis is also placed on motivation, which is considered the most important factor in the learning process. The motivation is referred as the first phase of a simplified model of the cognitive process, next phase is collecting their own experiences, clarification of the rules and relationships, inference of the knowledge and training of the knowledge.

The findings of J. Piaget or J. Dewey's theory is based of experiential learning (described by D. Kolb). The different phases of experiential learning are shown in figure 1.

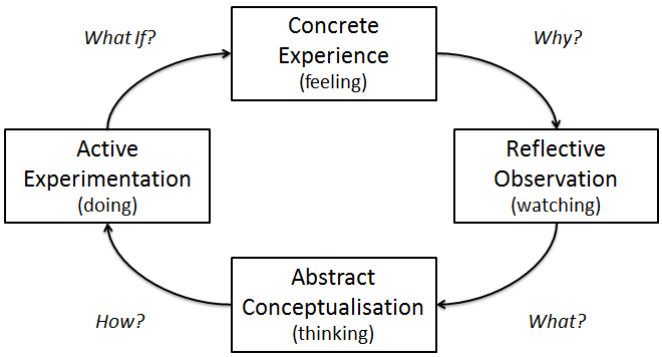


Fig. 1 Kolb's (Experiential) Learning Cycle.

Communicative approach is originally designed for teaching of foreign languages – language is not only understood as single structure (meaning the grammar and vocabulary), but language is also introduced and used in the sense of the executed functions. It was first introduced by G. H. Widdowson in 1978. One of the objectives of the communicative approach is to encourage the creativity of students. The communicative approach emphasizes on motivation and activity of students. Teacher is in position of guarantor of the methods and advisors. The motivational teaching methods like dialogical, problem or dramatic methods are used – see e.g. [10].

The Black Box method presents to students the task. Students investigated the relationship between cause (input) and the result (output). Student passes through three basic stages – see e.g. [11]:

- *abstraction* (student tries and watches what the is doing computer program and based on it derives general relationship);
- *encapsulation* (student considers how such a program could be written, composed its structure);

- *polymorphism* (based on previous experience student is able to describe in detail the connection between input and output).

These methods develop not only algorithmic thinking, but also a system approach, creativity and critical thinking of students. Block diagram of the Black Box Methods is shown on figure 2.

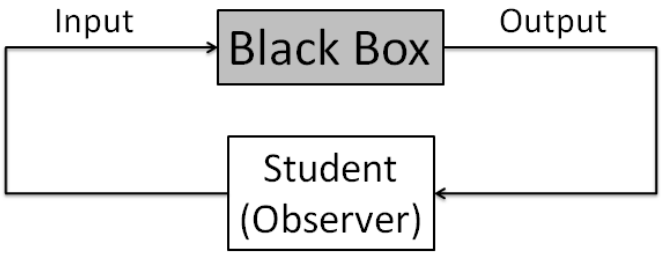


Fig. 2 Scheme of Black Box Method.

III. Algorithmic Thinking and Cryptology

Cryptology is the science that deals with the secrecy of messages.

We distinguish between three types of cryptology:

- *Cryptography* – creating form of ciphered messages that are understandable only to the addressee;
- *Cryptanalysis* - deciphering and breaking these ciphered messages;
- *Steganography* - creating of hide the message – the objective is to create the message that would not be found at all.

Some historical cryptosystems are known since antiquity (Caesar cipher or Hebrew ciphers). If we go through the history of cryptology, we observe how cryptology gradually improved and the procedures to encrypt and decrypt the text and techniques for solution of cryptograms are more.

However it is always necessary to use exactly specified sequence of steps to get the correct result. Therefore cryptology can be used for development of individual competencies of algorithmic thinking (which is nowadays considered to be one of the main key competencies of education from elementary school to high school).

A. Competencies of Cryptology in Algorithm Development

Within algorithmic thinking we can find these five competencies – see [12]:

- The ability to correctly apply the algorithm in a particular situation (student recognizes already known elements and applies known algorithm);
- Ability to create custom algorithms (analysis and re-synthesis of solutions to the sub-elements of the task);
- The ability to verify the accuracy and efficiency of the algorithm (to decide whether a given algorithm can be use in for given situation);
- The ability to recognize a problem that does not have algorithmic solution (they are tasks with no general algorithm of solution (e.g. Trisecting angle or square the circle), which is difficult to understand for many students)
- The ability to describe the algorithm in words.

B. Cryptology in Learning of Programming

Cryptology in learning of programming has been already presented in some publications.

Alternative learning of algorithm development using cryptology presents in his paper M. Capay [13]. He underlines mainly students' motivation and increasing of interest of student. These tasks are written in the form of mysteries that has to be detected. Algebraic riddles, puzzles and codes and ciphers are described on the principle of "black boxes". It is points out that the searched algorithm and programming solutions requires only basic knowledge of mathematical relationships. Ciphers are used for work with text strings. Only basic encryption techniques are used in the tasks. These tasks do not require explanation of its principle - namely mono-alphabetical substitution ciphers and transposition ciphers.

Another presented way of teaching of programming and algorithm development describes Š. Hubalovsky [14]. It is a systemic and multidisciplinary approach. Computer simulation of real systems based on knowledge of the situation in this field is created. The tasks are based on cryptanalysis - specifically cipher cryptanalysis mono-alphabetical cipher using bigrams [15] and trigrams [16] analysis of the reference text and the ciphertext. The solution is not without computer simulation hardly feasible.

The same approach is also engaged by M. Musílek. He describes case study of specific encryption method using Morse code – i.e. Morbit and Frationated Morse – see e.g. [17].

Encryption in the learning of algorithm development and programming has been used also by R. Morelli [18], which gives a detailed proposal for the project method in teaching of object-oriented programming. He describes particular task on historical encryption systems - Caesar cipher and simple transposition cipher.

IV. Case Study
Cryptology in Learning of Programming

A. Programming Language

Programming language Visual Basic for Application (VBA hereinafter), which is part of MS Excel (resp. MS Office) was selected in subject Programming in study program computer-aided teaching archiving. Advantage of programming in VBA in MS Excel is, that it allows work with MS Excel worksheets objects, mainly cells. The usage can be seen in deciphering using bigram [15] or trigrams analysis [16], where the process of storing the values into cells is very descriptive. Other advantage is possibility of creation of custom forms for easy control of the program. In addition, VBA programming

language based on Visual Basic (VB) is only adapted for MS Office products. The transition between these languages is due to the same syntax smooth.

B. Beginning Sample Lesson: the Exchange Values of Two Variables

Required knowledge: way of input of value and storage of value to variable (e.g. by using of InputBox ("the guiding text") and form of display the output (e.g. by using of MsgBox (X)).

Motivation:

Students have to solve the following situation: three containers are filled by different liquid. Two of them are filled by different liquids (or bulk materials). The task is to swap the contents of these containers. Props can be prepared for this activity – it depends on age of the students [10].

Example 1:

In the first practical tasks, students receive a new function in form of source code, which they have to describe verbally ("What happened when we used this source code?")

```
X = InputBox(„Input text including letter A“)
X = Replace(X, „a“, „u“)
X = Replace(X, „A“, „U“)
MsgBox(X)
```

The solution is the following description of the function:

```
Replace (expression, find, replace)
```

In the sequence of contiguous letters replace a certain substring with another substring.

The task may be accompanied by questions like:

- "What would happen if we missed a third line of code?";
- "What would be the output, if the letter “u” is replaced by other char in the second line of this code", etc.

Example 2:

In the next task students have to replace all the vowels by letter A (for example, when using language exercises, singing lessons). Students have to consider what letters are to be replaced and how the task can be simplified. Teacher can guide the students to use the function:

```
x = UCase (x).
```

This function replaces all letters to uppercase. This function is often used in encryption algorithms. The function can be also created by students using function:

```
Replace().
```

The task can be extended on removal all accents or punctuation.

The first task introduces the term function, the second task fixes the term function.

The next task develops algorithmic thinking of students (student can remember the motivational task which is help of solution).

Example 3:

Write program that exchanges the letters "A" and "E" in the text (i.e. if the input words is ABECEDA the output word will be EBACADE).

Students can use either the case-sensitive function (but at the end of the code all letters has to be changed to uppercase letters). Other and algorithmic better solution is based on using the third variable (another character), through which one of the specified letters is converted.

Example 4:

Write a program that will encrypt (or decrypt) text by Atbash cipher ("reverse alphabet" - the first letter is replaced with the last, the second is replaced with the second-last, etc.)

Example 5:

Write program that Morse code characters converted into the corresponding text.

This task is for some students quite difficult. Students have to order the commands Replace () from the longest to the shortest Morse alphabet letters. Example of bad sequence of commands:

```
…
X = Replace(X, "-/", "T")
X = Replace(X, "..-/", "U")
…
```

C. Final Project Lesson: Automatic Cryptoanalysis of the Cipher Text

Mastery of the principles algorithm development and programming is controlled by final project. Students use knowledge gained from previous lessons programming. Students are programmed in VBA.

The task of the project is to develop program for automatic cryptoanalysis of the monoalphabetical substitution cipher using frequency analysis of the bigrams of cipher text.

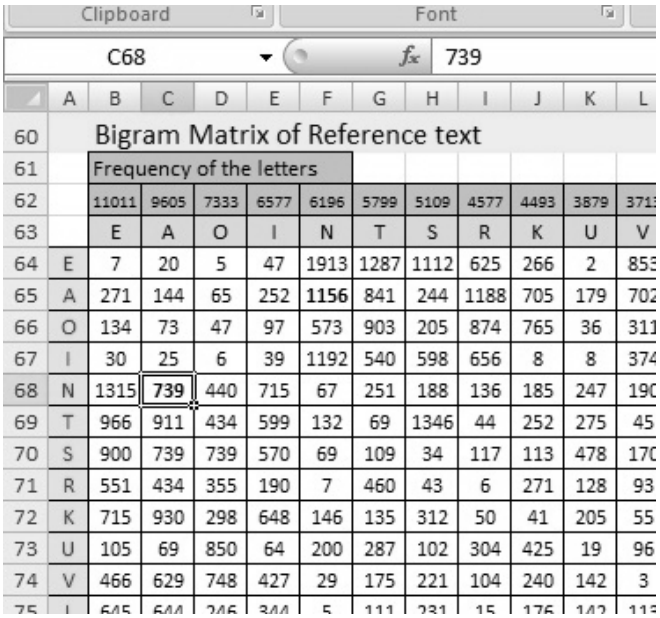
The solution of the above mentioned algorithmic task in MS Excel is subdivided into a number of more or less independent parts.

Detailed description of the automatic deciphering of monoalphabetical substitution cipher text can be found in [15].

1. Analysis of the reference text

The analysis of this reference text gives the first information about the frequency of individual letters, as well as the information of the frequency of bigrams of the selected language.

The final result of this step is creation of **bigram matrix *E* of reference text** – see figure 3.



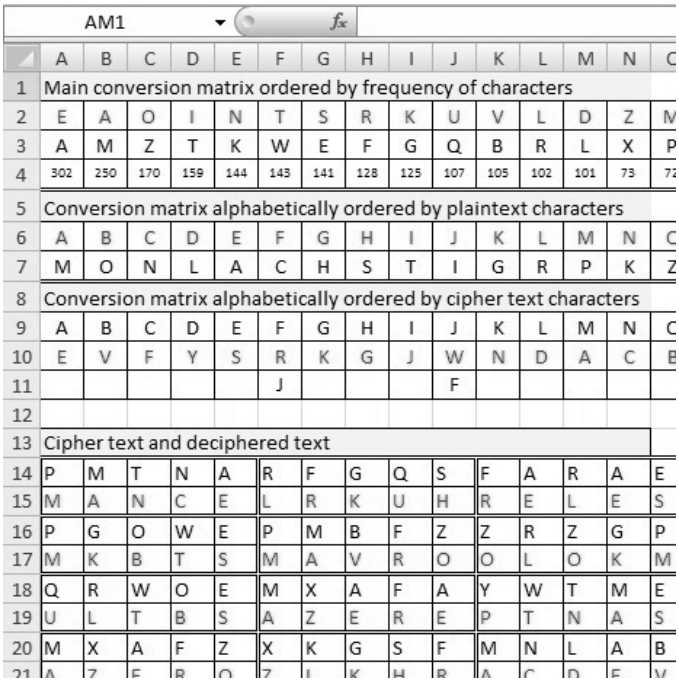
		A	B	C	D	E	F	G	H	I	J	K	L
60		Bigram Matrix of Reference text											
61		Frequency of the letters											
62		11011	9605	7333	6577	6196	5799	5109	4577	4493	3879	371	
63		E	A	O	I	N	T	S	R	K	U	V	
64	E	7	20	5	47	1913	1287	1112	625	266	2	853	
65	A	271	144	65	252	**1156**	841	244	1188	705	179	702	
66	O	134	73	47	97	573	903	205	874	765	36	311	
67	I	30	25	6	39	1192	540	598	656	8	8	374	
68	N	1315	**739**	440	715	67	251	188	136	185	247	190	
69	T	966	911	434	599	132	69	1346	44	252	275	45	
70	S	900	739	739	570	69	109	34	117	113	478	170	
71	R	551	434	355	190	7	460	43	6	271	128	93	
72	K	715	930	298	648	146	135	312	50	41	205	55	
73	U	105	69	850	64	200	287	102	304	425	19	96	
74	V	466	629	748	427	29	175	221	104	240	142	3	

Fig. 3 Bigram Matrix of Reference Text.

2. Downloading and analyzing of the cipher text

The next step is downloading and analyzing of the cipher text. In this step the cipher text has to be downloaded and corrected.

The final result of this step is creation of **frequency matrix of individual letters** (see figure 4) and creation of **bigram matrix *D* of cipher text**.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	C
1	Main conversion matrix ordered by frequency of characters														
2	E	A	O	I	N	T	S	R	K	U	V	L	D	Z	M
3	A	M	Z	T	K	W	E	F	G	Q	B	R	L	X	P
4	302	250	170	159	144	143	141	128	125	107	105	102	101	73	72
5	Conversion matrix alphabetically ordered by plaintext characters														
6	A	B	C	D	E	F	G	H	I	J	K	L	M	N	C
7	M	O	N	L	A	C	H	S	T	I	G	R	P	K	Z
8	Conversion matrix alphabetically ordered by cipher text characters														
9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	C
10	E	V	F	Y	S	R	K	G	J	W	N	D	A	C	B
11						J				F					
12															
13	Cipher text and deciphered text														
14	P	M	T	N	A	R	F	G	Q	S	F	A	R	A	E
15	M	A	N	C	E	L	R	K	U	H	R	E	L	E	S
16	P	G	O	W	E	P	M	B	F	Z	Z	R	Z	G	P
17	M	K	B	T	S	M	A	V	R	O	O	L	O	K	M
18	Q	R	W	O	E	M	X	A	F	A	Y	W	T	M	E
19	U	L	T	B	S	A	Z	E	R	E	P	T	N	A	S
20	M	X	A	F	Z	X	K	G	S	F	M	N	L	A	B

Fig. 4 Frequency Matrix of Individual Letters of Cipher Text.

In the end of this step the evaluation function *f* is calculated:

$$f = \sum_{i=1}^{26} \sum_{j=1}^{26} \left| D_{ij} - E_{ij} \right| \quad (1)$$

3. Analysis of bigrams of the cipher text and automatic deciphering

Analysis of bigrams of the cipher text and automatic deciphering of the cipher text is done in the final step.

The program analyses a text based on the frequency of the bigrams and create a new conversion matrix D' by a simultaneous substitution of two rows and two columns of the matrix *D*, calculates evaluation functions *f':*

$$f' = \sum_{i=1}^{26} \sum_{j=1}^{26} \left| D'_{ij} - E_{ij} \right| \quad (2)$$

and evaluates condition:

$$f' > f \quad (3)$$

If $f' > f$, the procedure continues with the next substitution of the letters in order. However, if $f' < f$, the procedure immediately stops and program suggest result of deciphering.

The process of automatic deciphering can be expressed by the flowchart shown in the figure 5.

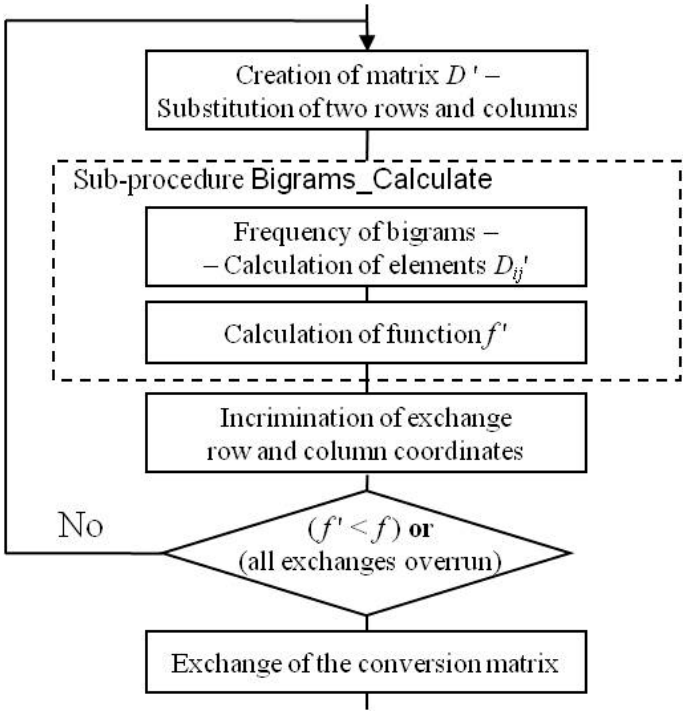


Fig. 5 Process of Automatic Deciphering.

V. RESEARCH

A. Research Methodology

The main research method is pedagogical experiment. The Dehnadi test for comparison of initial conditions in the experimental and control groups was used in the beginning of the research. The test investigates algorithmic thinking of the students and precondition of the students for the fulfillment the subject algorithm development and programming – see figure 6.

1. Read the following statements and tick the correct answer in the front column.	The new values of a and b are:

`int a = 10;`
`int b = 20;`
 	
`a = b;`
	☐ a = 30 b = 0
	☐ a = 30 b = 20
	☐ a = 20 b = 0
	☐ a = 20 b = 20
	☐ a = 10 b = 10
	☐ a = 10 b = 20
	☐ a = 20 b = 10
	☐ a = 0 b = 10
	☐ If none, give the correct values:
	a = b =

Fig. 6 Example of Dehnadi test.

B. Results of the Research

We found that there are several types of students in both experimental and control groups of student:

Type A: students handed blank test;

Type B: students to one question ticked more than one answer (the principle of the test is not understood);

Type C: students complete test correctly (60% of them already passed the course of programming in the previous study);

Type D: students complete test poorly;

Type E: students had some questions correct (in 90% they were the first three questions, which involve two variables, or even one question of three variables, when their value finally equal)

It is interesting that students who already attend course of algorithm development and programming belong to type C, D and E.

The distributions of students of experimental group are shown in figure 7 and distribution of student of control group is shown in figure 8.

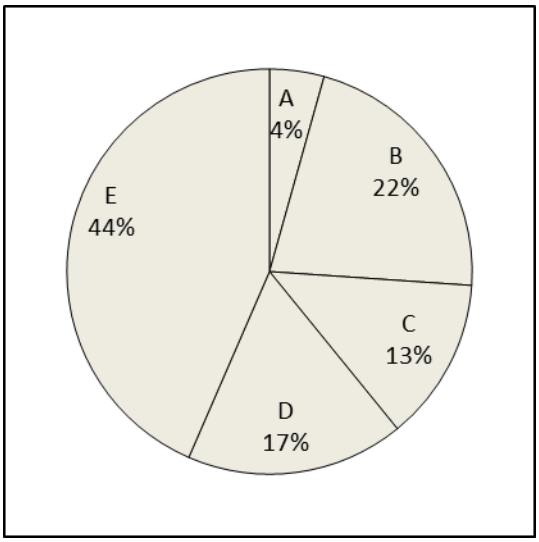


Fig. 7 Distributions of Students in Experimental Group.

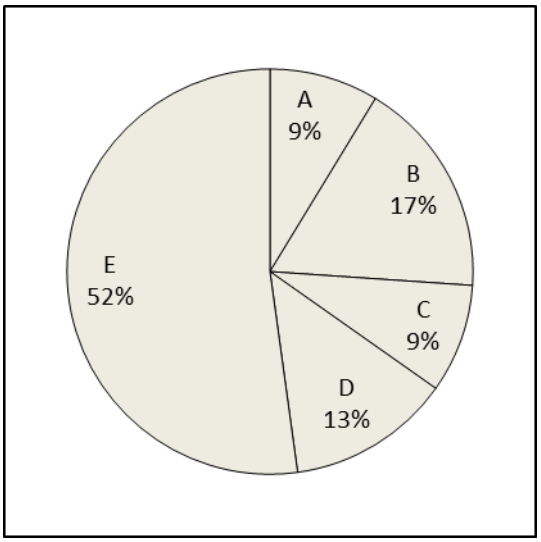


Fig. 8 Distributions of Students in Control Group.

The experimental group consist of 23% of girls, control group consist of 18% of girl.

The learning of algorithm development and programming in experimental group is carried out by proposed approach.

The learning in the control group is carried out by using classical, mostly math algorithms.

VI. Conclusion

Collection of exercises (tasks) for beginners of algorithm development and programming is created within the research. The tasks are focused on working with text - just basic knowledge of mathematics is required. The tasks are based motivational character of ciphers - it's something mysterious that is needed to be discovered. The ciphers and deciphering caused the development of computer technology, so the work with cipher and learning via cipher belongs to the field of information technology.

The methodology also uses various new approaches to teaching - constructivism, the principle of clarity, the technique of the Black Box or multidisciplinary approach.

The research described in the paper highlights the alternative learning of algorithm development and programming using cryptographic algorithms

The programming language Visual Basic for Application, which meets the requirements of modern programming language, were chosen in our research.

The final result of influence of using cryptology in learning of algorithm development and programming to development of algorithm thinking cannot be given without a final test. Based on observations and interviews with students in the experimental group, however, we can summarize these dependencies:

- Students who have input test without error, have no problems in seminars, they work independently and fulfilled given tasks.
- 80% of students who regularly attend seminars (75% of attendance) respond to the guiding questions and they are

ISBN: 978-1-61804-322-1

able to fulfill given tasks.

- Tasks related to text strings are for 50% of students more understandable than purely mathematical tasks (this group of students consists of 50% girls).
- Tasks related to historical cryptographic systems have motivational character. Students looked for various possible solutions and their programs were appropriately adjusted graphically and aesthetically. Their programs worked properly.

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] R. Pecinovky, "Současné trendy v metodice výuky programování", 2006 in *Proc. Počítač ve škole.* Nové město na Moravě: Available: http://www.vyuka.pecinovsky.cz/prispevky/2006PS_Soucasne_trendy_v_metodice_vyuky_programovani.pdf

[2] 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

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

[4] 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

[5] J. Oroma, Josephat O., T. Wikedzi, F. Ngumbuke, H. Wanga, "Algorithm First, Syntax Later Approach for Teaching Programming to Novices in Tanzanian Universitie: A Case of Tumaini University", in *Proc. EDULEARN12*, pp. 4259-4265, 2012

[6] M. Bilek, "*Konstruktivismus ve výuce přírodovědných předmětů*". [online]. Olomouc, Univerzita Palackého v Olomouci, 2008. [online]. Available: http://esfmoduly.upol.cz/publikace/bilek1.pdf

[7] L. Zormanova, "*Výukové metody v pedagogice: tradiční a inovativní metody, transmisivní a konstruktivistické pojetí výuky, klasifikace výukových metod*". Praha, Grada, 2012.

[8] L. Salanci, "Poznávací proces v školskej informatike", in Proc. *DidInfo 2013*, 2013. pp. 7-14 [online]. Available: http://didinfo.umb.sk/public/filestore/documents/richtext/153/zbornik_didinfo_2013.pdf

[9] M. Ben-Ari, "Constructivism in Computer Science Education", Journal *of Computers in Mathematics and Science Teaching*, Vol. 20, 2001, pp. 45-73.

[10] V. Vrbik, "*Komunikativní přístup k výuce programovacího jazyka*". Plzen, Západočeská univerzita, 2002.

[11] M. Lee, "*C++ Programming for the Absolute Beginner*", Boston, Course Technology, 2009.

[12] A. Jancarik, "*Algorithms and Solving Strategies*", Praha, Univerzita Karlova v Praze, 2007, [online]. Available: http://books.google.cz/books?hl=en&lr=&id=oif6CiJu1PsC&oi=fnd&pg=PA1&dq=info:1iLqXKaojiEJ:scholar.google.com&ots=kcrrIrVSBz&sig=HRHjBOW4zu7ZzV0W1r9Dc0B764c&redir_esc=y#v=onepage&q&f=false

[13] M. Capay, M. Magdin, "Alternative Methods of Teaching Algorithms", in: *Proc. Procedia - Social and Behavioral Sciences: 2nd World Conference on Educational Technology Research*, Vol. 83, 2013, pp. 431-436. Available: http://www.sciencedirect.com/science/article/pii/S187704281301152X

[14] S. Hubalovsky, "*Teorie systémů, modelování a simulace*". Hradec Králové, Gaudeamus, 2011.

[15] S. Hubalovsky, M. Musilek, "Automatic cryptoanalysis of the monoalphabetical substitution as a method of the system approach in the algorithm development thinking". *International Journal of Mathematics and Computers in Simulation*, Vol. 4, No. 4, 2010.

[16] S. Hubalovsky, P. Hanzalova, "Modeling, simulation and visualization of automatic cryptoanalysis of the short monoalphabetical substituted cipher text*"*. *International Journal of Mathematics and Computers in Simulation*. Issue 2, Vol. 7, 2013. p. 134-143.

[17] M. Musilek, "Morse Telegrah Alphabet and Cryptology as a Method of System Approach in Computer Science Education", in *Proc. DIVAI 2012 – 9th International Scientific Conference on Distance Learning in Applied Informatics*, 2012. p. 223-231, Available: http://conferences.ukf.sk/index.php/divai/divai2012/paper/view/859

[18] R. Morelli, R. Walde, G. Marcuccio. "*A java API for historical ciphers: an object-oriented design project*", *ACM SIGCSE Bulletin*, Issue 1, Vol. 33. New York, NY, USA, ACM, 2001, pp. 307-311.

Stepan Hubalovsky was born in Trutnov, Czech Republic in 1970. He obtained master degree in education of mathematics, physics and computer science in 1995 and doctor degree in theory of education in physics in 1998 both in Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic.

He worked 5 years as master of mathematics, physics and computer science on several secondary schools. He works as assistant 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.

Pavla Hanzalova obtained master degree in Mathematics and Music in Education in 2013 at University of Hradec Kralove. She is a student of doctoral degree in Information and Communication Technologies in Education at the University of Hradec Kralove. Her scientific activities are modelling, simulation and ciphers.