Volodymyr OVSYAK [1,2], Oleksandr OVSYAK [3], Mykhaylo KOZELKO [2], Julia PETRUSHKA [2]
[1] KIELCE UNIVERSITY OF TECHNOLOGY, 7 Tysiaclecia Panstwa Polskiego Av., 25-314 Kielce, Poland;
[2] UKRAINIAN ACADEMY OF PRINTING, 19 Pid Holoskom St., 79-020 Lviv, Ukraine;
[3] NATIONAL UNIVERSITY OF CULTURE AND ARTS, 5 Shuchevitsha St., 79-020 Lviv, Ukraine

An algebraic model of the subsystem for the computer generation of elimination operation

Abstract

A brief analysis of the known methods of non-algebraic and algebraic descriptions of algorithms has been presented. The features of the elimination operation being a term of the algorithm algebra have been shown. An algebraic model of the computer generation of the elimination operation has been synthesized by means of the algorithm algebra. The model has been implemented with the help of software on the platform Microsoft Visual Studio .NET. The models of subsystem as a graph and a diagram of sequences have been designed as well.

Keywords: Algorithm of algebra, algebraic methods, algebraic model of operation, elimination operation, graph of subsystem, diagram of sequences.

1. Introduction

There are known both non-algebraic and algebraic methods for describing of algorithms. Non-algebraic methods include verbal methods, block scheme ones, the method of recursive functions [1], the method of calculation λ [2], the method of Turing machines [3], the method of Post [4], Kolmogorov [5], Schönhage [6], the method of random access to the memory [7], the method of Markov's algorithms [8], the method of universal algorithms by Krinitski [9]. Algebraic methods include the system of algorithmic algebras by Glushkov [10, 11], the modified system of algorithmic algebras by Zeitlin [12, 13] and Pogorilyy [14], the primitive program algebra by Bui – Redko [15 – 20], the algebra of algorithms by V. Ovsyak [21] and the modified algebra of algorithms by O. Ovsyak [22, 23].

The use of algebraic [10 – 23] compared with non-algebraic [1 – 9] methods has certain advantages. In particular, models based on algebraic methods can be transformed using properties of algebraic operations to reduce costs of implementation. The method of mathematical induction can be used to do the research of the correctness of analytical models.

The system of algorithmic algebras and its modification are formed by the operations of conjunction, disjunction and inversion generalized on the three-symbol alphabet. These operations and the operation of forecasting are logical operations and they form the algebra of logics. The laws of excluded middle ($x \vee \neg x = 1$) and denial ($x \,\&\, \neg x = 0$), that occur in classical mathematical logics, are not performed in algebra of logics. Beside the algebra of logics, the system of algorithmic algebras and its modification include another operator algebra. The carriers of the operator algebra are the operators on which the operations of composition (multiplication), alternatives, cycle, filtering, asynchronous disjunction and synchronization are performed. The operation of composition is intended to describe the sequences of operators. The selection of the operators' implementation is described by the operation of alternatives. The operation of cycle is designed for cyclic repetition of the operators' implementation. To transform logical values into numerical ones on which the operators are executed, we will introduce the operation of filtering. The description of parallel processes is performed by the operation of asynchronous disjunction. The operation of synchronization is introduced for the time coordination of processes, which actually leads to time delays under certain condition.

By its definition [10 – 14], the operation of composition (multiplication) is not commutative but it is associative. The availability of the property of associativity of the operation of composition restricts its application solely to describe algorithms with permissible associativity of operators. However, algorithms are non-associative in most applications.

The signature of the primitive program algebra [15 – 20] is formed by the operations of superposition (substitution), branching and cycling. The carrier consists of partial n-arny functions on the universum and partial n-arny predicates in the universum, n = 1, 2, ... The description of sequences of partial functions and predicates is performed by the operation of superposition. The operation of superposition is associative. The availability of the property of associativity of the operation of superposition restricts the use of primitive program algebra. By means of the primitive program algebra, adequate models can be described only in associative algorithms and processes. However, non-associative algorithms and processes have the most practical dissemination.

The algebra of algorithms [21] and the modified algebra of algorithms [22, 23] include the operations of sequencing, elimination, paralleling, inversion and the operations of cyclic sequencing, cyclic elimination and cyclic paralleling. By these it means, the sequences in algorithms and processes are described by the operation of sequencing. In general, the operation of sequencing is non-associative. It adequately describes the classes both associative and non-associative algorithms and processes.

The operations of the algebra of algorithms and its modifications have specific graphic symbols of a complex shape. Such symbols are not included in typical mathematical symbols of operations. The operations are performed over operators. Geometric sizes of graphic symbols of the operations of elimination, sequencing and paralleling depend on the size of operators.

The most common computer editing software, for example, Word, can be used for automatic input and editing of formulas of the algorithm algebra and its modifications. Their use however requires considerable costs both programmer and computer working time. Processes of input and editing of formulas are difficult enough. In this context, the synthesis and implementation of mathematical models are relevant, including the subsystem of automatic generation of the graphic symbol of the operation of elimination for the given operators. This research is devoted to this issue.

2. An analytical model of subsystem decomposition

Described by the algebra of algorithms, the model of subsystem decomposition of automatic generation of the elimination operation for the given operator is presented by the formula (1).

$$pu\ @E{:}@T=\begin{cases}\overbrace{Z,E(),\ Cs(),\ Cfc(),\ Des(),Dra(),}\\ \underbrace{DraTex(),\ TaxHei(),\ TexLen(),\ FormTex(),\ VydEli().}\end{cases} \quad (1)$$

In the formula (1), $@E$ denotes the model of subsystem of elimination. The model of the subsystem $@E$ uses variable and functional operators of the subsystem $@T$, which we write as $@E{:}@T$, where: – is a symbol of affiliation (imitation) of the subsystem operators. $@E$ recorded on the left of the subsystem inherits the $@T$ recorded on the right of this symbol. Due to the fact that the subsystem of elimination must be accessible from other subsystems, then a general method of access (pu) is set to it,

which is standard and implemented by the keyword *public* [24, 25] in the software. The model of the subsystem includes: variables (*Z*), designed to store temporary data, functional operators setting initial values of variables (*E*()), calculating the size of the operators and the separator between them (*Cs*()), identifying the selection of elimination (*Cfc*()), de-selection (*Des*()), calculating the sizes and drawing the frame of the operation selection (*Dra*()), drawing the operator (*DraTex*()), calculating the length (*TexLen*()) and the height (*TexHei*()) of the operators and deleting (*VydEli*()) the operation of elimination. All operators are required, so the operation of sequencing must be applied for their description. But the place of relative position of operators, with the exception of variables, input of which must be made prior to their use, does not matter. The separator of operators is a coma, which is used to describe the properties of commutativity of operations in the algebra of algorithms.

3. A model of calculating the size of the operation of elimination

Geometric dimensions of the symbol of the operation of elimination depends on the size of the expression, which is located underneath. The expression is formed by three operators and separators between them. For accessibility of the functional operator from other subsystems, we make it generally accessible with an opportunity of re-definition (*ov*) of its definition in this subsystem, which is described in the imitation subsystem. Let the input parameters are *dv*∈*@DraV* type of the subsystem *DraV*, which is realized by the known class *DravingVisual* [24, 25] and *f*∈*@Siz* type of the subsystem *Siz*, which is realized by the known class *Size* [24, 25], and the functional operator itself is described by the formula (2).

```
pu ov Cs(dv∈@DraV,f∈@Siz)=
    sepSiz∈@Siz=@Siz(TexLeen(;),TexHei(;))
        ;
    wid=sepSiz.Wid*2+f.Hei3
        ;
    hei=sepSiz.Hei
        ;
    tA.Cs(dv, f); * ; (tA≠S)-?
        ;
    wid=wid +tA. wid
        ;
    hei=tA.hei; * ; (hei<tA.hei)-?
        ;
    tB.Cs(dv, f); *: (tB≠S)-?
        ;
    wid = wid +tB. wid
        ;
    hei =tB. hei; * ; (hei <tB. hei)-?
        ;
    con.Cs(dv, f); * ; (con≠S)-?
        ;
    wid = con. wid
        ;
    hei = con. hei; * ; (hei < con. hei)-?
        ;
    hei = hei +Mat.Sqr(wid)/2+2
    ;
    W
    ;
    (ori=OriHor)-?
```

(2)

In the formula:

- *sepSiz*∈*@Siz* = *@Siz*(*TexLen*(;), *TexHei*(;)) – the operator of calculating the length is *TexLen*(;)
- *TexHei*(;) – is the operator of calculating the height of the separator (;) of the operators of elimination and their attribution to the variable *sepSiz* of the type *Siz*;
- *wid* = *sepSiz.Wid**2 + *f.Hei**3 – is the increase of the length of the expression under the symbol of the operation of elimination with regard to the lengths of the two separators of operators and the height of the selected font *f.Hei**3;
- *hei* =*sepSiz.Hei* – is the attribution of the value of the height of the separator *sepSiz.Hei* to the variable of the height of the expression *hei*;
- *tA*≠$)-? – is the description of the verification of the availability of the first operator on the left (*tA*) of the operation of elimination;
- *tA.Cs*(*dv*, *f*) – is the selection of the functional operator *Cs*(*dv*, *f*) of the subsystem, the operator (*U*) and calculating the length of the first operator;
- *wid*=*wid*+*tA.wid* –is the increase of the total length of the expression under the symbol of the operation of elimination with regard to the length of the first operator;
- (*hei*<*tA.hei*)-? – is the comparison of the current value of the height of the expression with the height of the first operator;
- *hei*=*tA.hei* – is the attribution of the value of the height of the first operator to the variable of the current height of the expression (when the value of the current height is less than the value of the height of the first operator);
- (*tB*≠$)-? – is the verification of the availability of the second operator on the left (*tB*) of the operation of elimination;
- *tB.Cs*(*dv*, *f*) – is the selection of the functional operator *Cs*(*dv*, *f*) of the subsystem operator (*U*) and calculating the length of the second operator *tB*;
- *wid* = *wid* + *tB.wid* – is the increase of the total length of the expression under the symbol of the operation of elimination with regard to the length of the second operator;
- (*hei*<*tB.hei*)-? – is the comparison of the current value of the height of the expression with the height of the second operator;
- *hei* = *tB.hei* – is the attribution of the value of the height of the second operator to the variable of the current height of the expression (when the value of the current height is less than the value of the height of the second operator);
- (*con*≠$)-? – is the verification of the availability of the conditional operator;
- *con.Cs*(*dv*, *f*) – is the selection of the functional operator *Cs*(*dv*, *f*) of the subsystem operator (*U*) and calculating the length of the conditional operator *con*;
- *wid* = *con.wid* – is the increase of the total length of the expression under the symbol of the operation of elimination with regard to the length of the conditional operator;
- *hei*<*con.hei*)-? – is the comparison of the current value of the height of the expression with the height of the conditional operator;
- *hei* = *con.hei* – is the attribution of the value of the height of the conditional operator to the variable of the current height of the expression (when the value of the current height is less than the value of the height of the conditional operator);
- *hei* = *hei* + *Mat.Sqr*(*wid*)/2 + 2 – is the calculation of height of the symbol of the operation of elimination, taking into account the ratio between the length and height;
- (*ori* = *Ori.Hor*)-? – is the verification of the value of the variable of orientation of the operation of elimination;
- *W* – is the formula that describes calculating the size of the symbol of the operation of elimination for its vertical orientation and it includes the operators similar to the operators of horizontal orientation of the operation. It is described by formula (3):

$$
\begin{aligned}
W= \; & wid=sepSiz.\underline{Wid} \\
& ; \\
& Hei=sepSiz.\underline{Hei}*2+f.\underline{Hei}+16 \\
& ; \\
& \overline{tA.Cs(dv,\ f);\ *\ ;\ (tA\neq S)-?} \\
& ; \\
& \overline{wid=tA.wid;\ *\ ;\ (wid<tA.wid)-?} \\
& ; \\
& hei=hei+tA.hei \\
& ; \\
& \overline{tB.Cs(dv,\ f);\ *:\ (tB\neq S)-?} \\
& ; \\
& \overline{wid=tB.wid;\ *\ ;\ (wid<tB.wid)-?} \\
& ; \\
& hei=hei+tB.hei \\
& ; \\
& \overline{con.Cs(dv,\ f);\ *\ ;\ (con\neq S)-?} \\
& ; \\
& \overline{wid=tB.wid;\ *\ ;\ (wid<tB.wid)-?} \\
& ; \\
& hei=hei+con.hei \\
& ; \\
& wid=wid+\underline{Mat}.\underline{Sqr}(hei)/2+2
\end{aligned}
\qquad (3)
$$

4. A graph of subsystem of elimination

The model of the subsystem of elimination is shown as a graph in Fig. 1.

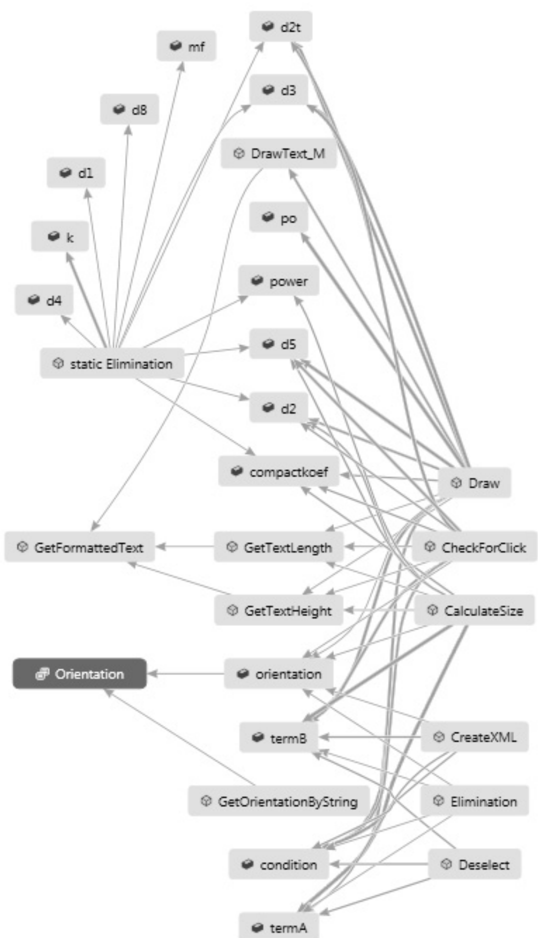

Fig.1. Graph of the subsystem "*Elimination*"

It contains the functional operators:

- *Elimination* – setting the values of variables,
- Orientation – a set of values of the operation orientation,
- *GetOrientationByString* – the recognition of the given operation orientation,
- *CalculateSize* – the calculation of the size,
- *CheckForClick* – the selection of the operation,
- *Draw* – drawing the symbol,
- *Deselect* – deleting the selection,
- *CreateXML* – the formation of the textual description,
- *DrawText_M* – setting the parameters of formatting the values of the operation components,
- *GetTextHeight* – the calculation of the components height,
- *GetTextLength* – the calculation of the length of the operation components,
- *GetFormattedText* – formatting the value of the operation components,
- *Delete* – the removal of graphic figures, and all other components are variable.

5. Fragments of the diagram of sequences

Fragments of the diagram of sequences of the functional operators for an automatic calculation of the sizes of the operation of elimination are shown in Fig. 2a), b) and c).

Fig. 2a) shows the selection (●⟶) of the functional operator for calculating the size *CalculateSize* of the operation of elimination. It is called *CalculateSi…* in the diagram and it is located in the subsystem *this:Elimination*. It comprises a selection (*GetTextLength*) of the functional operator (*GetTextLength(";", f)*) for calculating the length of the separator of operators. The separator is the symbol ";". We use the value of the font size (*f*) of the typeface to calculate the value of the length.

Next there is a selection (*GetFormatedText*) of the functional operator (*GetFormatedText(text, f, Height)*) for formatting the calculated length of the separator. Similarly, it comprises a selection (*GetTextHeight*) of the functional operator, the calculation (*GetTextHeight(";", f)*), the selection of the functional operator (*GetFormatedText*) for formatting and the formatting (*GetFormatedText(text, f, Height)*) of the calculated height of the separator of the operators of the operation of elimination.

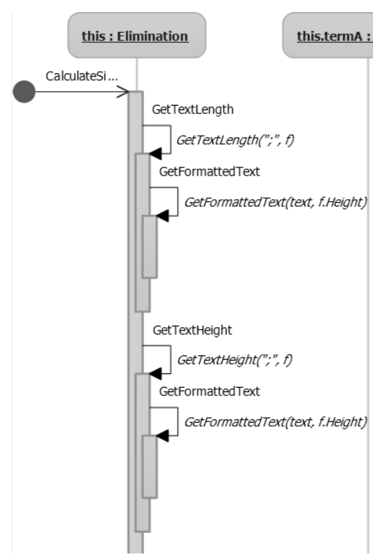

Fig. 2 a) Beginning of the diagram of sequences

The fragment of the diagram of sequences (Fig. 2b) shows the verification (*[if(collapsedflag==0)]*) of the availability under the symbol of the operation of elimination of the expression. If the

expression is the operation of sequencing, then we can detect the availability of the horizontal orientation (*[if(orientation == Orientation.Horizontal)]*). If the orientation is horizontal, then we can detect the availability of the first operator (*[if(termA != null)]*) of the operation of elimination. The availability of the operator leads to the selection (*CalculateSize*) of the functional operator for calculating its size (*termA.CalculateSize(f)*). Similarly, as it is in the fragment of the diagram of sequences (Fig. 2 a), we select and perform the functional operators for calculating the length (*GetTextLength*, *GetTextLength(separatorString, f)*), and the height (*GetTextHeigth*, *GetTextHeigth(separatorString, f)*), and formatting (*GetFormatedText*, *GetFormatedText(text, f, Heigth)*), and (*GetFormatedText*, *GetFormatedText(text, f, Length)*) of the first operator of the operation of elimination.

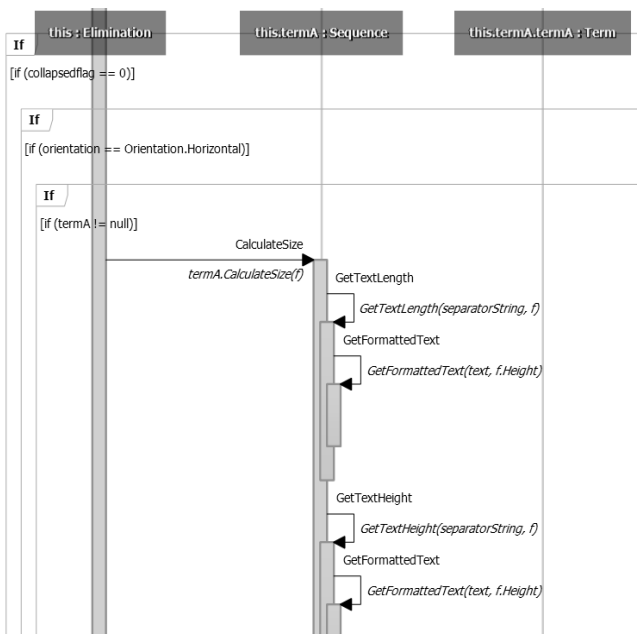

Fig. 2 b) Continuation of the diagram of sequences

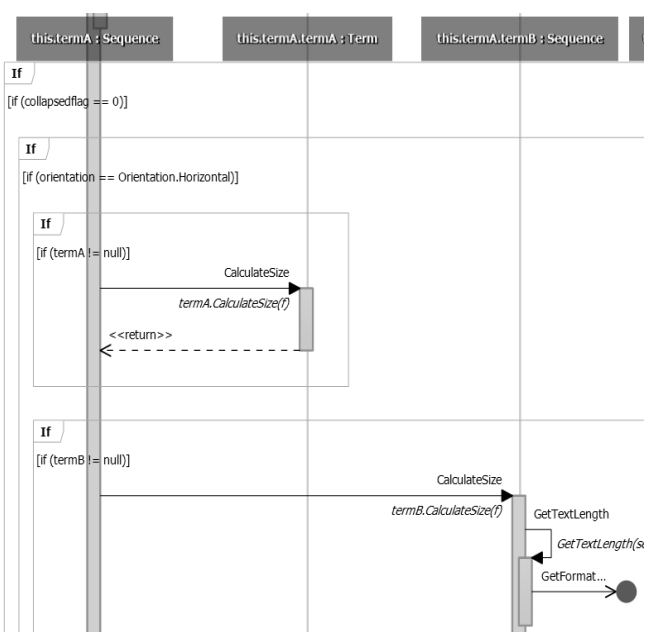

Fig. 2 c) Fragment of the diagram of sequences for calculating the size of operators of the operation of elimination

Fragment of the diagram (Fig. 2c) shows conditions of availability of the expression under the symbol of the operation of sequencing (*[if(collapsedflag == o)]*) of the horizontal orientation of the operation (*[if(orientation == Orientation.Horizontal)]*), of the first elementary operator (*[if(termA != null)]*) and the second operator (*[if(termB != null)]*) as a sequence. The fulfilment of the condition *[if(termA != null)]* is accompanied by the selection (*CalculateSize*) and the calculation of the size of the elementary operator by the functional operator *termA.CalculateSize(f)* which is located in the subsystem *Term*. For the second operator of the sequence –we make the selection (*CalculateSize*) and the calculation of the size (*termB.CalculateSize(f)*) by the functional operator, which is located in the subsystem *Sequence*.

6. Conclusions

1. Algebraic models, compared with non-algebraic ones, are more compact and more accurate and the equivalent transformation can be made with them for the optimization according to the selected criteria.

2. The following methods for describing algorithms are well known: the system of algorithmic algebras, the modifications of the system of algorithmic algebras and the primitive program algebra. The methods are adequate tool for the description of associative algorithms and processes.

3. The algebra of algorithms and its modification provides getting adequate models of both associative and non-associative algorithms and processes.

4. Synthesized models describe the subsystem of automatic generation of the operation of elimination for the given adequate operators.

7. References

[1] Kleene S.C.: Origins of Recursive Function Theory. Annals of the Theory of Computing. 1981, vol. 3, no. 1, pp. 52–67.
[2] Church A.: An Unsolvable Problem of Elementary Number Theory. American Journal of Mathematics, 1936, vol. 58, pp. 345–363.
[3] Turing A.M.: On Computable Numbers, with an Application to the Entscheidungsproblem. Proc. of LMS, 1936, ser. 2, vol. 42, no. 3, 4, pp. 230–265.
[4] Post E.L.: Finite Combinatory Processes – Formulation I. JSL, 1936, vol. 1, no. 3, pp. 103–105.
[5] Kolmogorov A.N.: On the Definition of Algorithm (in Russian). Uspekhi Mat. Nauk, 1958, vol. 13:4, pp. 3-28 (translated into English in AMS Translations 29, pp. 217–245, 1963).
[6] Schönhage A.: Storage Modification Machines. SIAM Journal on computing, 1980, vol. 9, no. 3, pp. 490–508.
[7] Aho A.V., Hopcroft J.E., Ullman J.D.: The Design and Analysis of Computer algorithms. Addison-Wesley Publishing Company, 1974.
[8] Markov A.A., Nagorny N.M.: The Theory of Algorithms (Mathematics and its Applications), 2001, Springer.
[9] Krinitski N.A.: Algorithms around Us. Moscow, Nauka, 223 p. (in Russian), 1984.
[10] Glushkov V.: Automata Theory and Formal Transformations of Microprograms. Cybernetics, 1965, no. 5, pp. 1–9 (in Russian).
[11] Glushkow W.M., Zeitlin G.E., Justchenko J.L.: Algebra. Sprachen. Programmierung. Berlin: Akademie-Verlag, 1980, 340 p.
[12] Zeitlin G.E.: The Algebraic Algorithmics: The Theory and Applications. Cybernetics and Systems Analysis, 2003, no. 1, pp. 8–18 (in Russian).
[13] Zeitlin G.E. Elements of Algebraic Algorithmics and Algorithmic Oriented Synthesis of Parallel Programs. Mathematical Machines and Systems, 2003, no. 2, pp. 56–67 (in Russian).

[14] Pogorilyy S.D.: A Conception for Creating a System of Parametric Design for Parallel Algorithms and Their Software Implementations. Cybernetics and System Analysis, 2009, vol. 45, no. 6, pp. 952–958.
[15] Redko V.N.: Primitive Program Algebras. Cybernetics, 1984, no. 4, pp. 1–7 (in Russian).
[16] Redko V.N.: Primitive Program Algebras of Functions of Rational Arguments and Values. Reports of the Academy of Sciences of the USSR. Ser. A. Physical and mathematical and engineering sciences, 1986, no. 6, pp. 65–67 (in Russian).
[17] Redko V.N.: Primitive Program Algebras of Integer and Lexical Functions. Reports of the Academy of Sciences of the USSR. Ser. A. Physical and mathematical and engineering sciences, 1984, no. 10, pp. 69–71 (in Russian).
[18] Redko V.N.: Primitive Program Algebras of Functions, which Arguments and Values are Vectors, Relations and Matrix. Reports of the Academy of Sciences of the USSR. Ser. A. Physical and mathematical and engineering sciences, 1987, no. 7, pp. 3–5 (in Russian).
[19] Redko V.N.: Primitive Program Algebras of Computable Functions. Cybernetics, 1987, no. 3, pp. 68–74 (in Russian).
[20] Gubskiy B.V.: Primitive Program Algebras of Computable Functions and Predicates on Relations and Tables in the Final and Countable Alphabets. Reports of the Academy of Sciences of the USSR. Ser. A. Physical and mathematical and engineering sciences, 1988, no. 10, 78–79 (in Russian).
[21] Ovsyak V.K.: The Tools for Equivalent Transformations of Algorithms of Information-Technology Systems. Reports of National Academy of Ukraine, 1996, no. 9, pp. 83–89 (in Ukrainian).
[22] Owsiak W.: Rozszerzenie algebry algorytmów. Pomiary, Automatyka, Kontrola, 2010, no. 2, pp. 184–188.
[23] Owsiak A.: Algebraic Models of Subsystems of Abstract System with the User Interface. Pomiary, Automatyka, Kontrola, 2013, no. 11, pp. 1179–1182.
[24] Nagel C., Evjev B., Glynn J., Watson K., Skinner M.: C# 2012 and .NET 4.5. 2013, Wiley Publishing, Inc.
[25] Powers L., Snell M.: Microsoft Visual Studio 2010. 2011.
[26] Ovsyak V.: A sequential method for the synthesis of formulae of algorithms. Measurement Automation Monitoring, Jan. 2015, vol. 61, no. 01, pp. 21–23.

Received: 09.03.2018 *Paper reviewed* *Accepted: 04.05.2018*

Prof. Volodymyr OVSYAK, DSc, eng.

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling. He works as an professor in Kielce University of Technology.

e-mail: ovsyak@rambler.ru

Prof. Oleksandr OVSYAK, DSc, eng.

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling. He works as an associate professor in National University of Culture and Arts.

e-mail: ovsjak@ukr.net

Mykhaylo KOZELKO, MSc, eng.

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling.

e-mail: actionmike@mail.ru

Julia PETRUSZKA, eng.

She specializes in applied computer science, theory of algorithms, programming, information systems, modeling of systems, computer simulation and mathematical modeling. She works as an graduate student in Ukrainian Academy of Printing.

e-mail: julja-petrushka@rambler.ru