SYSTEM MODIFICATION MODEL CREATING SYNTHESIS FOR ALGORITHM FORMULAS

Magdalena Niziołek, Volodymyr Ovsyak

***Abstract**: An analysis of the basic system for the synthesis of formulas of algebra algorithms was conducted The need for a construction of an expanded system for computer synthesis that would allow simplifying the notation of algebra formulas was demonstrated. Basing on functional criterion purpose a three-level model of decomposition was created. The model is recorded with the help of formulas of algebra algorithms. An implementation of the created model was made in the MS Visual Studio 2010 platform in C# programming language and samples are presented.*

***Keywords**: Algebra, algorithm, model, formula, subsystem, operations: formatting editor*

***ACM Classification Keywords**: F.2 Analysis of algorithms and problem complexity*

Introduction

The basic editor allowed operations only over two trivial uniterms. For more advanced algorithms the large amount of symbols led to surplus of form over content. A necessary for a new, advanced editor occurred. A new model of a computer system was created. As next, based on that model an actual editor was build. The editor assist in writing the algorithm's algebra formulas, saving it and loading exiting formulas as well as editing the data. It automatically compute the places for new elements, thus users don't need to manually correct the formulas, as it has place in the standard editors.

An example of the Euclidean algorithm written as block-diagram and as Ovsyak's formula written in the new editor is as given (Figure 1).

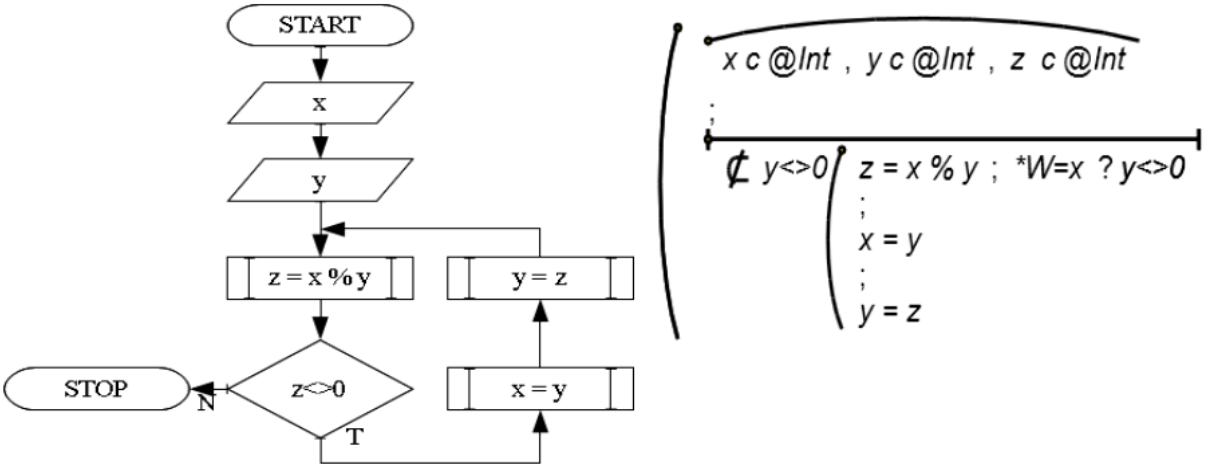

Fig. 1. An egzample of Euclidean algorithm in form of Ovsyak's formula

Where x, y are non-negative integers, z contains value of the modulo operation and *W is output message.

Computer system for creating formulas

Describing algorithms in the form of algebra forms allows not only the advantage of the formal description, it allows also to simplify transformation an algorithm into an actual computer data, thus an easier automatic optimization.

The new formula editor is written in Microsoft Visual Studio .NET 2010 with use of the modern languages XAML and C#. The actual algorithm is created and visible in the main editor in its graphic form, where a code that is easy for interpretation for a computer is at the same time written as an XML document. A description of creating complex uniterms was described in [Ovsyak, Niziołek, 2011], [Niziołek,Ovsyak, 2011]. This paper will describe the creation of operation in the above mentioned system.

Model

All models are shown as formulas of Ovsiak's algebra. The system's main model (@G) is as follow (Figure 2). The explanation of the operations and the symbols of the algebra are given in [Ovsyak, 2008].

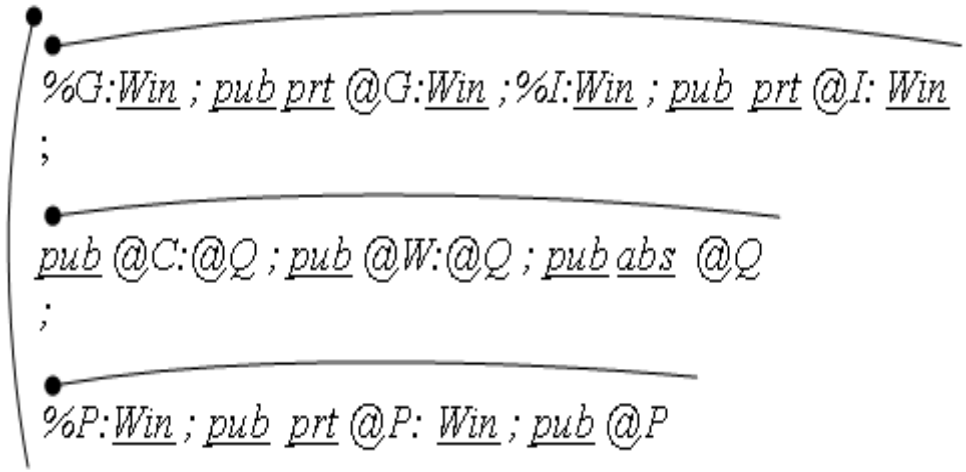

Fig. 2. Formula of the systems main model

where G is the main system, that is composed of two parts @G and %G, where @ indicate the functional subsystem and % indicate the graphic part of the system. The subsystem of creating complex uniterms is composed of @Q an abstract subsystem for terms, @W a subsystem for creating single uniterms, and @C that creates the complex uniterm. %I is its graphic part. Subsystem for creating operations is composed of @Q and %P.

The model of subsystem responsible for creating operations has the following formula (Figure 3), where:

- all the uniterms are given access method pub, which implements a well-known definition as public, with the exception of DrwCnt that has access method prv (private),
- Sprtr, Orntn and Drct are variables from the standard subsystem, enum holds information about the operation's separator, orientation and direction

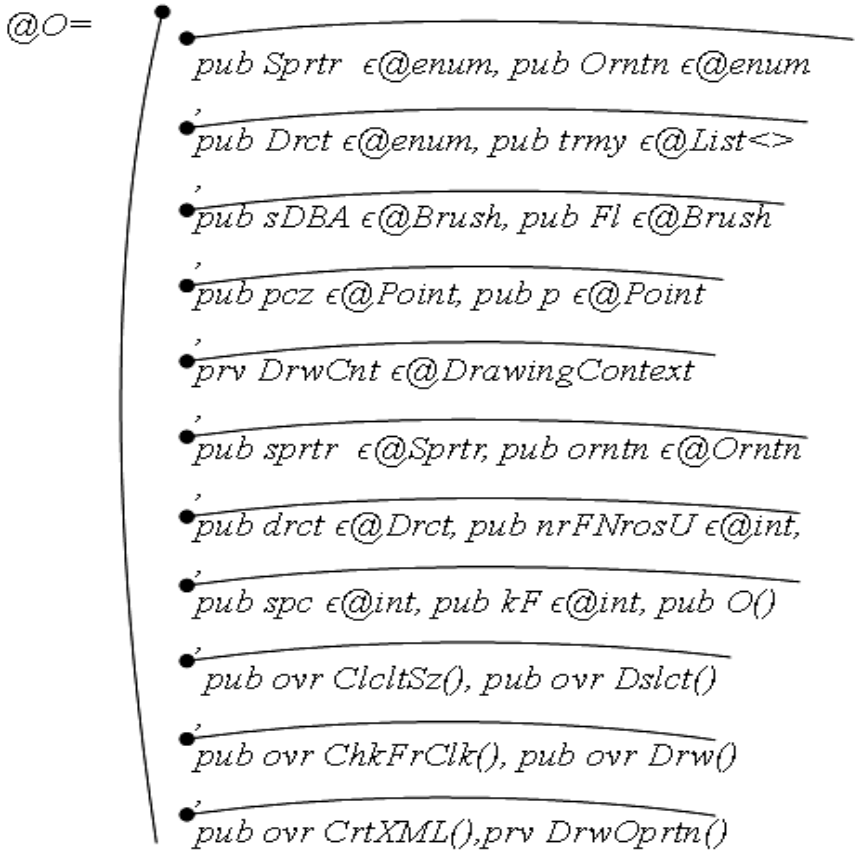

Fig. 3. Form of the subsystem for creation operations

- trmy variables from the standard subsystem List is a list of all uniterms that the operation will contain
- sDBA and Fl - variables from the standard subsystem Brush contains information about the main and second brush, that is used to paint the actual operation's symbol
- pcz and p - variables from the standard subsystem Point are uniterms that contain the beginning and current coordinate of where the operation should be placed in the main window
- DrwCnt - variable from the standard subsystem DrawingContext
- sprtr, orntn and drct - variables from the above mentioned enum type and are part of the operation that held information about the separator symbol, orientation and direction of the selected object
- nrFNrosU, spc and kF - variables from the standard subsystem Int and store information needed for creating the operation's object
- O() is the constructor, that create given operation
- ClcltSz() calculates the height and width for the new operation
- Dslct() deceselect the operation if it was chosen in the main window
- ChkFrClk() check if the operation was chosen
- Drw() draw the borders to indicate the choose of the operation, as well as others elements that are needed
- DrwOprtn() draws the specific symbol of the chosen operation
- CrtXML() creates a XML document basing on the editor nodes.

As an example of the functional uniterm a part (due to it's actual length) of the model of Drw() is shown (Figure 4).

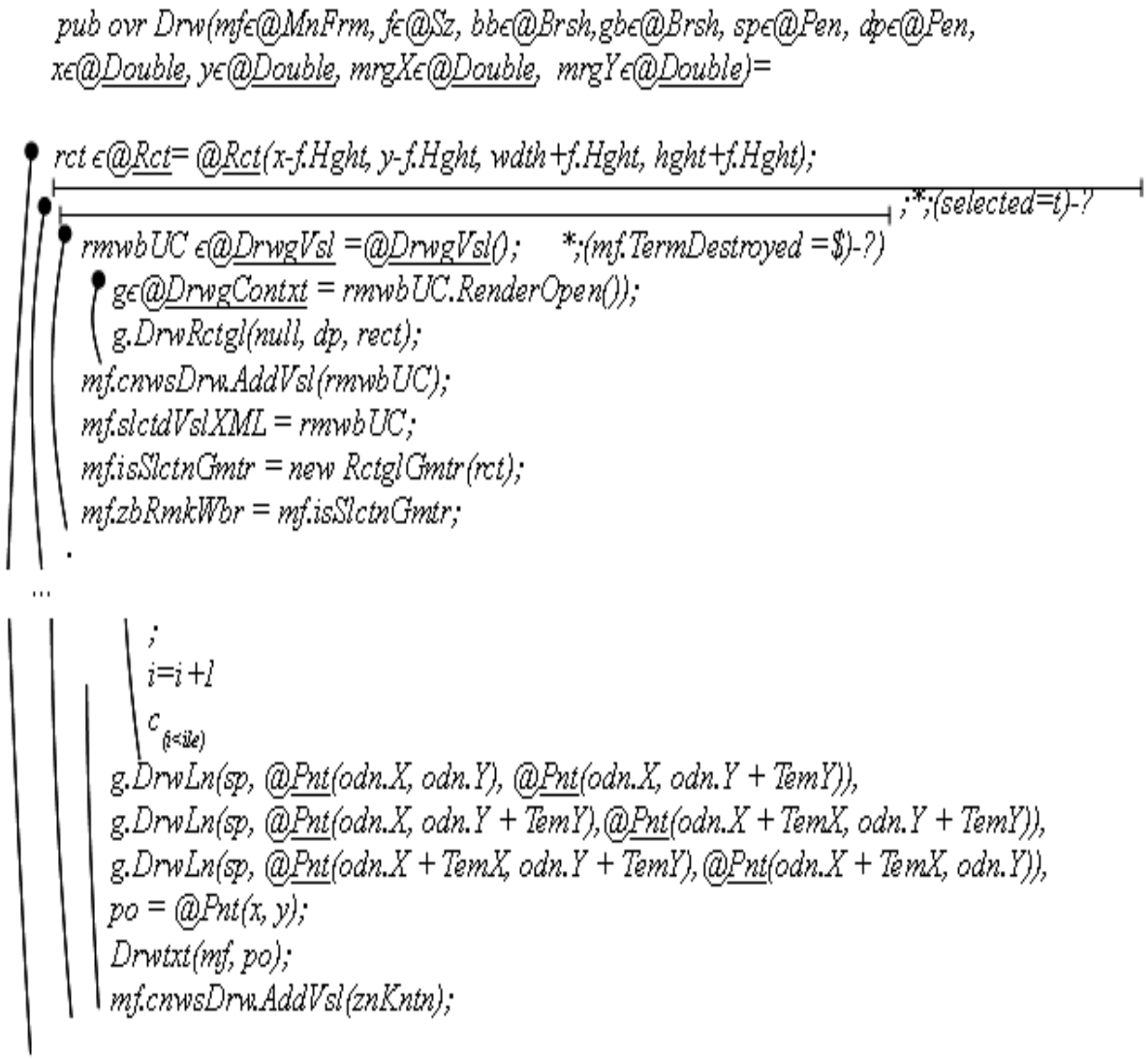

Fig. 4. The beginning and end of an functional uniterm

The model contain uniterms from the subsystem @Q and system @G, as well the standard systems.

Implementation

The main system contains classes that are system classes like MyCanvas or Form and those that were created based on the system model, like Uniterm or Sequence (Figure 5).

The model for operations is implemented for three classes: sequence, elimination and parallelization, as well partially for their cyclic counterparts. The class diagram for sequence is as follow (Figure 6).

Fig. 5. Classes used in implementation of the system

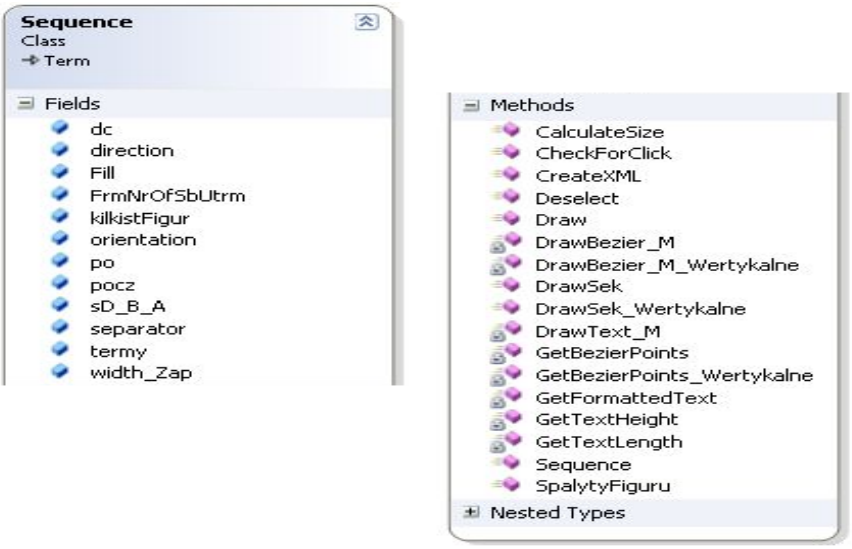

Fig. 6. Implementation of the operation's subsystem in the form of a class for creating sequences

To add an operation to the formula one has to choose an object in the main window and as next the operation symbol. The object (uniterm or operation) will be converted to the requested operation. A subsystem is called that call a graphic window for forming an operation. Next the constructor Sequence() is called and creates a new object with chosen parameters. The functional uniterm ChkFrClk() examine what part of the form was chosen. If it was uniterm, simple operation or complex operation. After receiving the information Dslct() is called. It deselect the chosen object. As next the functional uniterm ClcltSz() is calculating the size that the new operation will need in the main window (that is the size of all uniterms, separators and the operator sign size). The Drw() functional uniterm is drawing requested uniterms and call on the DrwOprtn(), that draws the requested operation symbol. At the end DrwCnt() redraws requested contents in the main window. The CrtXML() functional uniterm is called when the user saves his work.

Example

The editor showed has a tool menu divided in three sections. The first section shows button for the mostly used operations (Figure 7), that is sequencing, elimination, parallelization and their cyclic counterparts (the red frame). The second section show button for manipulating the data: creating complex uniterms, adding correct etc. of the data, connecting to database. The third section allows to customize the editor area.

To start working with the editor an uniterm or operation is need to be selected. Adding an operation to the formula is intuitively easy, by choosing the requested option from the main menu or toolbox menu. A new window will appear, that contains parameters for the new operation (Figure 8).

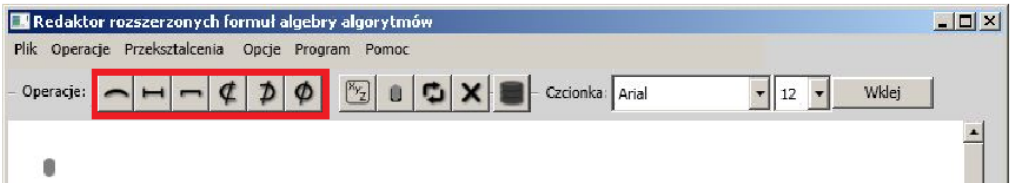

Fig. 7. Main window of the implemented system for creating Ovsyak's formuls.

Fig. 8. Windows (part of subsystem) for setting parameters for operation

The parameters are as following. The orientation of the operation can be horizontal or vertical, the separator can be a coma or a period. Accepting the parameters new operation is added to the editor **(Figure 9).** Now only data in the uniterms are need to be added.

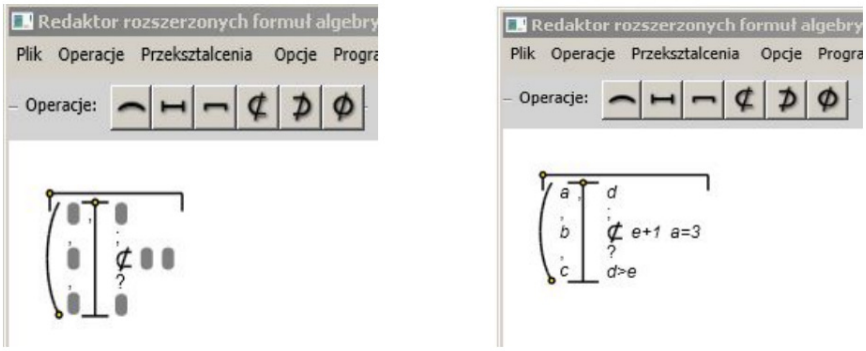

Fig. 9. Operations in the editor: a) without data, b) with data

The XML allows easy saving and loading the formula into the editor. Because of it's popular format it can be also easy imported and edited in another editors, like notepad, if the required structure is being preserved.

The example gives the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<root>
 <parallelisation direction="beginning" separator="comma" orientation="horizontal">
  <sequence direction="ending" separator="comma" orientation="vertical">
   <uniterm nr=”0”>a</uniterm>
   <uniterm nr=”1”>b</uniterm>
   <uniterm nr=”2”>c</uniterm>
  </sequence>
  <elimination direction="beginning" separator="semicolon" orientation="vertical">
   <uniterm nr=”0”>d</uniterm>
   <cyclic-sequence orientation="horizontal">
    <uniterm nr=”0”>e+1</uniterm>
    <uniterm nr=”1”>a=3</uniterm>
   </cyclic-sequence>
   <uniterm nr=”2”>d&gt;e</uniterm>
  </elimination>
 </parallelisation>
</root>
```

The change in the notation between the basic and the extended editor are visible (Figure 10). In the extended editor only one symbol is needed for the same operation over more than two uniterms. The dot indicated witch uniterm shall be considered as a first. The extended editor has also an option for reducing the number of the symbols, that could appear while editing the algorithm or after optimization. At the same time the XML file is extended to cover the additional information.

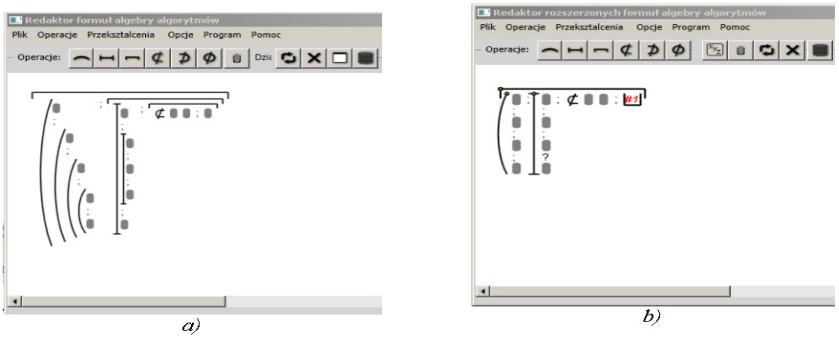

Fig. 10. Example of forms in: a) basic, b) the extended editor

Conclusion

Using algebra of algorithms allows algorithms to be described like a mathematical formula. The operations inform about connection between the parts. To help creating algorithm formulas a computer system was build, that allows easy edition of the formula and help to create computer friendly data for further use, in example optimization of the algorithm.

Bibliography

[Shannon, 1949] C.E.Shannon. The Mathematical theory of communication. In: The Mathematical Theory of Communication. Ed. C.E.Shannon and W.Weaver. University of Illinois Press, Urbana, 1949.

[Ovsyak, 2008] Ovsyak V.K. Computation models and algebra of algorithms. Informatsiyni systemy ta merezhi. Visnyk Natsionalnoho universytetu “Lviska politekhnika”. 2008. vol 621 - pp.3-15.

[Niziołek, Ovsyak, 2011] Niziołek M., Ovsyak V.: A concept of a model of a computer system for forming complex uniterms. ITHEA IBS ISC No.: 24, Rzeszow, Poland; Sofia, Bulgaria, 2011, pp. 35-39.

[Ovsyak, Niziołek, 2011] Ovsyak V., Niziołek M.: Modele składowe unitermów złożonych. Pomiary, automatyka, kontrola, 2011,vol 9. - S. 1090-1092.

[Post, 1936] Post E. L., Finite Combinatory Processes - Formulation 1. Journal of Symbolic Logic, 1936, vol 1, pp. 103-105,.

[Turing, 1936] Turing A. M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of London Mathematical Society, series 2, vol. 42 (1936-1937), pp. 230-265;

[Kolmogorov, 1958] Kolmogorov A. N., Uspensky V.A.: On the definition of algorithm. Uspekhi Mat. Nauk 13:4 (1958), pp. 3-28

[Schönhage, 1970] Schönhage A.: Universelle Turing Speicherung. In J. Dörr and G. Hotz, Editors, Automatentheorie und Formale Sprachen, Bibliogr. Institut, Mannheim, 1970, pp. 369-383.

[Aho, 1974] Aho A.V, Hopcroft J.E, Ullman J.D.: The design and analysis of computer algorithms. Addison-Wesley Publishing Company, 1974.

[Markov, 2001] Markov A.A., Nagorny N.M.: The Theory of Algorithms (Mathematics and its Applications). Springer, 2001.

[Church, 1936] Church A.: An unsolvable problem of elementary number theory. American Journal of Mathematics, vol. 58 (1936), pp. 345-363.

[Kleene 1981] Kleene S.C.: Origins of recursive function theory. Annals of the Theory of Computing, 1981, vol. 3, pp. 52-67.

[Krinitski, 1988] Krinitski N.A.: Algorithms around us. Mir, Moscow, 1988

[Gluschkov, 1980] Gluschkov W.M., Zeitlin G.E., Justchenko J.L.: Algebra. Sprachen. Programmierung. Berlin: Akademie-Verlag, 1980–340 p.

[Cejtlin, 1998] Cejtlin G.E. Vvedenije v algoritmiku. K. Sfera, 1998. pp. 310

[Doroszenko, 2003] Doroszenko A.E., Cejtlin G.E. Algebroautomatnyje specyfikacji paralelnych program nad obszczej i rapredelonnoj pamiatiu /Problemy programuvannia, 2003, vol 3, pp.5-21.

[Cejtlin, 2008] Cejtlin G.E., Zakharija L.M. Algebraicheskije aspekty polnoty: abstrakciji, biologia i ekologia /Problemy programuvannia. 2008, vol 2-3, pp.31-36

Authors' Information

Magdalena Niziołek *– Opole University of Technology, Faculty of Electrical, Control and Computer Engineering, ul. Sosnkowskiego 5, 45-271 Opole, Poland; e-mail: m.niziolek@doktorant.po.opole.pl*

Major Fields of Scientific Research: programming, theory of algorithms,

Volodymyr Ovsyak *– Opole University of Technology, Opole, Poland and Ukrainian University of Printing, L’vov , Ukraine ; ovsyak@rambler.ru*

Major Fields of Scientific Research:Theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling.