Velbitskiy I. V.
The Glushkov's Fund, Kyiv, Ukraine, ivelbit@gmail.com

Graphical programming with only one scheme (arc) without statements of type goto, if, for,... without labels and brackets of type begin-end, {-},... or 5.6, 55.2 and 15.4 advantages

Two methods for writing of any program in the single graphic shell of the next generation are proposed for all existing programming languages [1–5], Fig. 1 and 2.

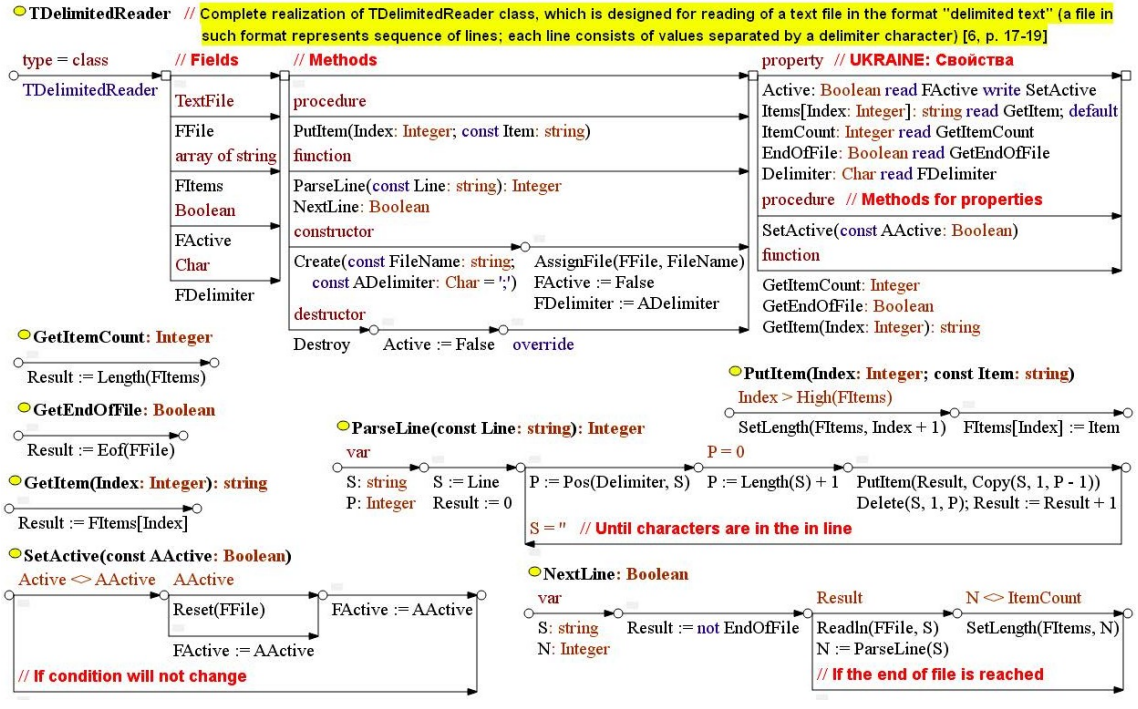

Figure 1. Graphics program is *5.6-fold more compact* and incomparably *more visual* compared to its traditional writing [6, p.17-19]

For the first time it is possible to represent a structure (idea, architecture) of the programming project using only one scheme (arc) and without realization details as if a "bird's-eye view" to the entire project, Fig. 2. This simplifies development, discussion, protection, demonstration and project maintenance. Transition from the first write to the second and back is made with one functional key.

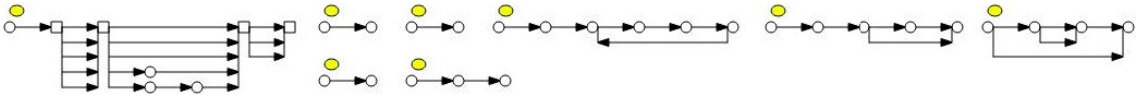

Figure 2. Structure of the same program without realization details is *55.2-fold more compact*

In addition to mentioned advantages, this program allows starting the project development "from idea, architecture, layout" with gradual building up and developing a skeleton with details and realization capabilities. The work can be executed in any concept (OOP, COP etc.) and in any language from which only the simplest, linear objects and concepts of type of procedures, functions, expressions, formulas, data structures etc. and established in classical mathematics are used. Entry of such program into a machine is 15.4-fold faster. Just 37 (number of horizontal arcs) clicks with a mouse button are required to enter the program given in Fig. 2. More than 500 filler words (see the

graphical scheme in Fig. 1) were deleted from traditional form of this program writing in Delphi [6].

The next generation graphic shell comprises of the schemes. Each scheme is assigned the name written near its yellow ellipse, Fig. 1. An elementary scheme (so-called R-scheme) from which any scheme (program) is built in any language comprises of one horizontal arc between two nodes, Fig. 3, 1–2. The arc has any direction (to the right, to the left, without direction). Condition for passing through the arc is written above the arc, and executed action is written below the arc. Any number of arcs may radiate from one node to the left and/or to the right, Fig. 4, 1–2. Any scheme is constructed from elementary R-schemes and it is used along the entire life cycle of the program and a programmer.

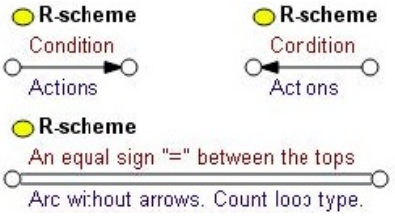

Figure 3. Elementary R-scheme (one arc) to the right, to the left and without an arrow for writing any program in any programming language

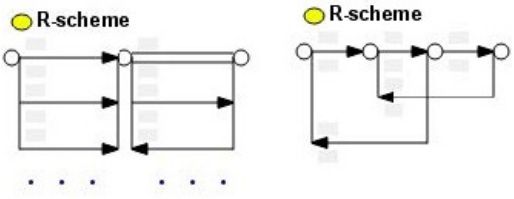

Figure 4. Examples for condition writing without restrictions, cycles of type "daisy wheel" and "Olympic rings" that are not available in traditional programming languages

Therefore, symbolic statements of type *goto*, *if*, *case*, *for* are the great "evil" of the current programming. As for today, they are too much 1) oriented to a computer (setting of tasks to a computer) rather than to a human (provision of convenience and quality of human's cognitive processes at realization of specific task); 2) they are primitive in terms of capabilities; 3) they have complex syntax; 4) they use a plenty of excessive filler characters such as «;», labels, key words *then*, *else*, *do*, *continue*,...brackets of type *begin-end*, {-}; 5) they do not make possible to document thoughts, motivation of the programmer's actions upon program development, which led to necessity to use second additional language of diagram UML in the development process, and 6) they require to restate conditions of the task that has to be programmed. Therefore, they make in general simple process of programming difficult and confusing as well as they create a situation when "nothing is seen behind the trees in the forest", i.e. the idea of algorithm and program is not discovered. Such things do not exist in graphical programming, which is closer to classic mathematics, it is more visual, simpler, more powerful, and 5-50 folds more compact compared to traditional program writing. The new programming concept allows proving correctness of the programs [5], uniting the programmers and making their programming language with free syntax, and teaching to programming from the age of 5-7 years and many other things, which begin with the words "for the first time" and "no" in traditional programming. *At present, this is just the top of the iceberg of advantages of graphical programming with only one R-scheme (with one arc).*

Non-commercial realization of the system for C++ is given on the website of the Glushkov's Fund www.glushkov.org Developers – programmers: Anton Khodakovskiy and Alexander Gubov.

References. **1.** V.M. Glushkov, I.V. Velbitskiy, Programming Technology and Automation Problems, USIM, Kyiv, №6, 1976, p. 75-93. **2.** I.V. Velbitskiy, Programming Technology, Tekhnika, Kyiv, Ukraine, 1984, 279 p. **3.** INTERNATIONAL STANDARD ISO/IEC 8631. Information technology-Program constructs and convention for their Representation – Second edition 1989.08.01 Geneva 20, Switzerland: ISO/IEC Copyright Office, p. 7, 1989. **4.** McHenry William K. R-Technology: A Soviet Visual Programming, Journal of Visual Languages and Computing,vol.1,#2,p.199-212,1990 **5.** I.V. Velbitskiy, Graphical Programming and Program Correctness Proof. (Plenary report), 9th IEEE COMPUTER SCIENCE & INFORMATION TECHNOLOGIES CONFERENCE, Armenia, Yerevan, 23-27 Sept. 2013, IEEE Xplore, CSIT-2013. **6.** A.N. Valvachev, K.A. Surkov, D.A. Surkov, Yu.M. Chetyrko "Programming in Delphi INTERNET, Chapter 3 OOP, 19 p., 2005

Authors' information