PROCEEDINGS

of the Union of Scientists - Ruse

Book 5

Mathematics, Informatics and Physics

Volume 8, 2011

RUSE

ISSN 1311-9974

The Ruse Branch of the Union of Scientists in Bulgaria was founded in 1956. Its first Chairman was Prof. Stoyan Petrov. He was followed by Prof. Trifon Georgiev, Prof. Kolyo Vasilev, Prof. Georgi Popov, Prof. Mityo Kanev, Assoc. Prof. Boris Borisov, Prof. Emil Marinov. The individual members number nearly 300 recognized scientists from Ruse, organized in 13 scientific sections. There are several collective members too – organizations and companies from Ruse, known for their success in the field of science and higher education, or their applied research activities. The activities of the Union of Scientists – Ruse are numerous: scientific, educational and other humanitarian events directly related to hot issues in the development of Ruse region, including its infrastructure, environment, history and future development; commitment to the development of the scientific organizations in Ruse, the professional development and growth of the scientists and the protection of their individual rights.

The Union of Scientists – Ruse (US – Ruse) organizes publishing of scientific and popular informative literature, and since 1998 – the "Proceedings of the Union of Scientists- Ruse".

BOOK 5

"MATHEMATICS, INFORMATICS AND PHYSICS"

VOLUME 8

CONTENTS

BOOK 5

"MATHEMATICS, INFORMATICS AND PHYSICS"

VOLUME 8

Education

Participation in International Projects

AN EDUCATIONAL TOOL FOR NOVICE PROGRAMMERS

Galina Atanasova, Katalina Grigorova

Angel Kanchev University of Ruse

***Abstract**: The paper investigates the characteristics of novice programmers, their difficulties in introductory programming courses. There are described variety aspects of programming – too many reasons for novices' overwhelming and de-motivation. The significant role of visualization environments and algorithm animation tools is presented. It is underlined that the provision of a useful and accurate mental model and algorithm making abilities will positively influence a novice's success in programming. A tool design for algorithm animation and its learning aids are presented.*

***Keywords:** Computer Science, Algorithm Animation, Algorithm Visualization Tools, Novice Programmers*

INTRODUCTION

Data structures and algorithms are important foundation topics in computer science education. Students deal with algorithms in many computer science courses. For instance, in computer graphics, students learn objects rendering algorithms, in networking, they study algorithms that solve networks track congestion, and in database, they learn algorithms that search or sort data. Accordingly, teaching algorithms is a common activity that takes place in many computer science classes. However, algorithms are often hard to understand because they usually model complicated concepts, refer to abstract notions, describe complex dynamic changes in data structures, or solve relatively difficult problems. Consequently, teaching algorithms is a challenging task that faces instructors and requires a lot of explaining and illustrating. Therefore, teaching aids other than conventional are needed to help students understand algorithms better [6]. The ability to realize graphic representations faster than textual representations led to the idea of using block schemes to describe the behaviour of algorithms to learners.

DIFFICULTIES IN LEARNING PROGRAMMING

Learning to program is generally considered hard, and programming courses often have high dropout rates [10]. Educational research has been carried out to recognize the characteristics of novice programmers and to study the learning process and its connections to the variety aspects of programming. Let explore these issues more closely.

CHARACTERISTICS OF NOVICE PROGRAMMERS

By definition, novice programmers lack the knowledge and skills of programming experts. Several different separating factors have been studied in the literature and were also reviewed by Robins et al. [11]. Common features for novices seem to be that they are limited to surface knowledge of programs and generally approach programming "line by line", rather than at the level of bigger scope. Novices spend little time in designing and testing their algorithm. When necessary, try to correct their mistakes with small local fixes instead of more thoroughly reformulations [2]. The knowledge of novices tends to be context specific rather than general [3], and they also often fail to apply correctly the knowledge they have obtained. In fact, an average student does not usually make much progress in an introductory programming course [2]. This was also noticed by the study of McCracken et al. [5], who noticed serious deficiencies in student's programming skills in introductory courses. This supports the empirical observations of many teachers; programming novices often fail to recognize their own deficiencies. Also the personal

properties of the students affect their performance. Mathematical or science abilities seem to be related to success at learning to program [4, 7]. In an introducing course, different student behaviours in confronting a problematic situation can be recognized. Perkins [9] named two main types novices: stoppers and movers. In problematic situation stoppers simply stop and abandon all hope of solving the problem on their own, while movers keep trying, modifying their algorithm and use feedback about errors effectively. There are also extreme movers, "thinkers", who cannot track their work, make changes more or less randomly, and like stoppers do not progress very much in their task. There are effective and ineffective novices, i.e. students who learn without excessive effort and those who do not learn without inordinate personal attention [11]. Naturally, students' personal learning strategies and motivation affect their success in learning programming strategies. Robins et al. [11] stated that "Given that knowledge is (assumed to be) uniformly low, it is their pre-existing strategies that initially distinguish effective and ineffective novices". Prior knowledge and practices can also be a major source of errors, especially when trying to transfer a step-by-step problem-solving solution directly from a natural language into a program [13]. The differences between the natural language and a programming language could easily cause problems.

VARIETY ASPECTS OF PROGRAMMING

Learning programming contains several activities, e.g., learning the language features, algorithm design, and algorithm comprehension. Typical approach in textbooks and programming courses is to start with declarative knowledge about a particular language. However, studies show that it is important to bring also other aspects to the first programming courses. Several common deficits in novices' understanding of specific programming language constructs are presented in Soloway and Spohrer [13] and collected also by Pane and Myers [8]. For example, variable initialization seems to be more difficult to understand than updating or testing variables.

However, the main source of difficulty does not seem to be the syntax or understanding of concepts, but rather basic algorithm planning [11]. It is important to distinguish between programming knowledge and programming strategies [1]. Winslow [14] noticed that students may know the syntax and semantics of individual statements, but they do not know how to combine these features into valid algorithms. Even when they know how to solve the problem by hand, they have trouble translating it into an equivalent algorithm.

Students have often great difficulties in understanding all the issues, relating to the execution of a program. Students have difficulties in understanding that each instruction is executed in the state, which has been created by the previous instructions.

There is often little correspondence between the ability to write a program and the awareness to read one. Programming courses should include them both. In addition, some basic test and debugging strategies should be taught [14]. Robins et al. [11] suggest that one more issue that complicates the learning of programming is the distinction between the mental model as it was intended, and the program model as it actually is. There are often mistakes in the design and bugs in the code. Also in working life, programmers face daily the need to understand a program that is running in an unexpected way. This requires an ability to track code to build a mental model of the program and predict its behavior. This is one of skills that could be developed by emphasizing algorithm comprehension and debugging strategies in the programming courses.

We can generalize that it is clear that novices are burdened by having to learn so many new things in introductory programming courses. This leads to an overwhelming perception of incapability and uphill struggle in a large proportion of novices. Their self-

esteem may be getting defeated before they have even reached base subjects. Whilst syntax might be a difficulty for novices, it is evident that the more pressing issue is the deficiency in their problem solving and algorithm construction skills. Novices can understand the individual concepts of programming in isolation but have significant difficulties putting them together in order to express a problem's solution. The provision of a useful and accurate mental model and algorithm making abilities will positively influence a novice's success in programming.

THE ROLE OF ALGORITHM ANIMATION AND VISUALIZATION TOOLS IN NOVICE PROGRAMMERS INTRODUCTORY COURSES

Algorithm visualization and animation tools are designed to make visible aspects of programming often hidden from the programmer. As such they are capable of promoting "low-level" models of algorithm features such as models of execution. They can reinforce a model of an algorithm execution by explicitly showing how the execution of a statement affects the program state and hence the environment in which the following statement is executed.

Computerized aids that reinforce mental models and reliable algorithm construction of the programming process are far more appropriate for novice programmers than commercial environments [12]. There are many environments aimed novice programmers learning improvement. These tools provide some benefit, but it is clear that these ideas can be improved upon. If the features of these tools were usefully enhanced, extended and combined, it might prove to be more beneficial to novice programmers.

FLOWCHARTS

Flowcharts have traditionally been used to visualise algorithm structures and are a very appropriate visualisation form for novices. They are easy to learn, can be easily understood with little or no prior training and provide the novice with an accurate mental model of an algorithm and its components. Furthermore, flowcharts aid the processes of abstracting a problem into a solution and the transition between problem specification and syntactical solution.

Studies of flowchart-based animation tools have shown that by animating a flowchart, we can further its effectiveness and offer a concrete model of execution by demonstrating the interaction between algorithm components.

Our work benefits are directed to algorithm animation and block scheme advantages, usefully extend and combine in an application with learning aids. We suppose that in this way we will decrease variety problems in introductory programming courses and diminish the mental models level of abstraction.

A TOOL DESIGN FOR ALGORITHM ANIMATION

This research presents a novice-programming tool. It is aimed at facilitating the imperatives first approach to teaching introductory programming. This tool improves on existing approaches by combining multiple forms of animation with algorithm variables visualization features and the manually animation of flowcharts. It allows the user to construct a wide range of algorithms, involving variables, assignment, decisions, and looping. It is these basics that underpin the core problem solving aspects of imperative programming. However, knowledge of these concepts provides important skills, useful for developing process logic in any programming paradigm. The designed tool focuses on using flowcharts to develop visual solutions to basic programming problems. This provides the user with an accurate mental model of the algorithm structures. It provides the facility to animate its algorithms and visualises the effect each algorithm statement has on any

variables. The animation features and interaction with the visual representation reinforce student understanding of both the visual solution and algorithm statement flow. The tool allows the user to focus on the problem solving aspects of algorithm constructing whilst, minimising all other distractions. This has been achieved by eliminating the necessity of writing complex and confusing syntax and reducing the learning curve associated with professional development environments. These distractions cloud the problem solving processes at the heart of programming; minimising their impact reduces the cognitive overload that inhibits the algorithm construction abilities of many novices.

Our tool's user interface is simple and has a main working area of visualization for the flowchart design. There is learner support, providing for block context depended information entering. This tool's feature ensures user's mistake avoidance. The remaining areas of its user interface contain the controls to construct, animate, save and load flow charts, as well as perform other useful commands.

A flow chart creation

In the tool users can create an algorithm by interacting only with the flowchart and do not have to trouble themselves with entering large amounts of complex and confusing syntax. To add a component to a program, the user simply selects it from a component toolbar and clicks on the relevant part of the flowchart. The component is defined and if successful, gets added below the component selected (shown in figure 1). If the user makes an error during the component definition, they are presented with an accurate and meaningful error message.

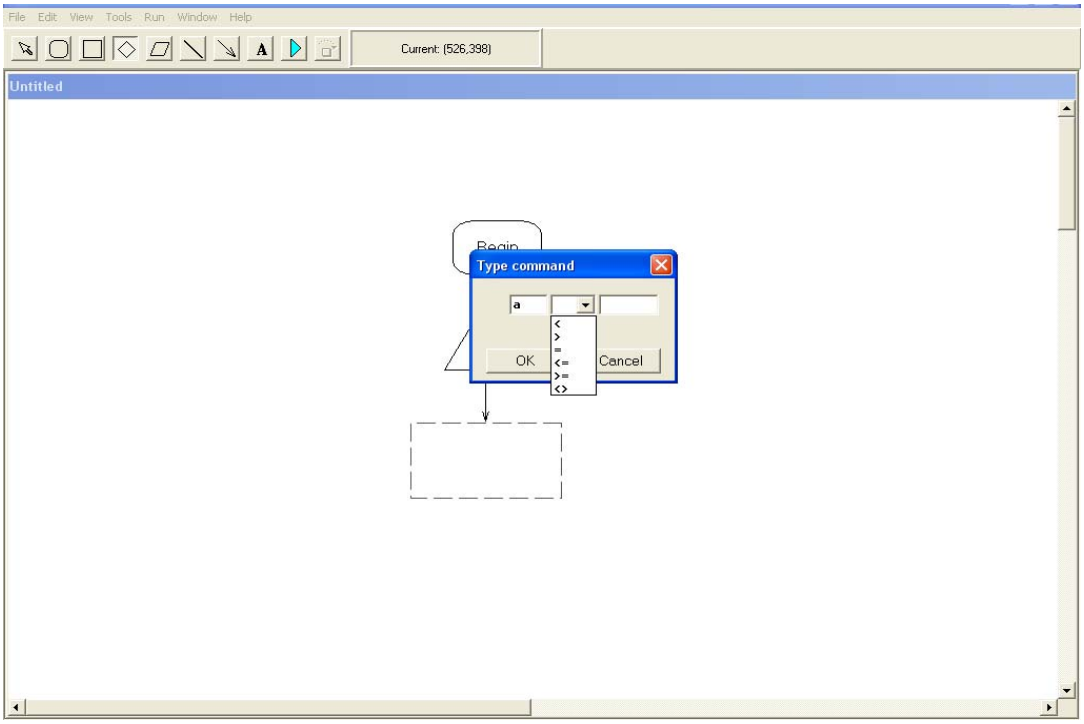

Figure 1. The Tool's User Interface

Variable handling

The values of algorithm variables are visualised in the window "Variables". This window is visible during algorithm execution/animation the variable inspector allows the user to observe in real time the effect each algorithm statement has on the data used.

Algorithm animation

The tool improves on the capabilities of the static flowchart by animating its

flowcharts to show an algorithm in action. The animation features emphasise algorithm flow concepts and the interaction between algorithm components in the flowchart representation. Combining this with the variable inspection features provides the user with an accurate and concrete model of a working algorithm.

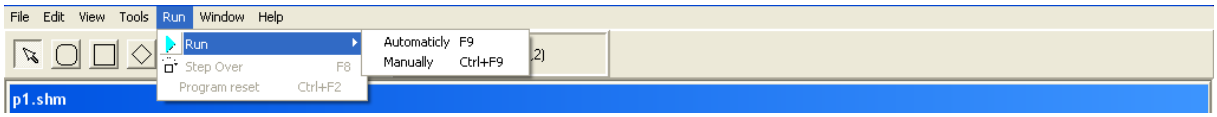

Figure 2. User Opportunities for Algorithm Visualization

The learner has two opportunities to visualize the algorithm action: automatically and step by step - manually (figure 2). In the first mode only the input variables' values are entering and the final result is shown (figure 3). This mode provides a feature for quick algorithm correctness testing. The step by step animation of a flowchart is achieved by tracing through the flow of an algorithm, highlighting each flowchart component and viewing relevant effect of its execution in real time by the variables' window. When a cyclic or conditional construction is reached, the flow in algorithm is diverted appropriately, based on the result of the logical expression it contains.

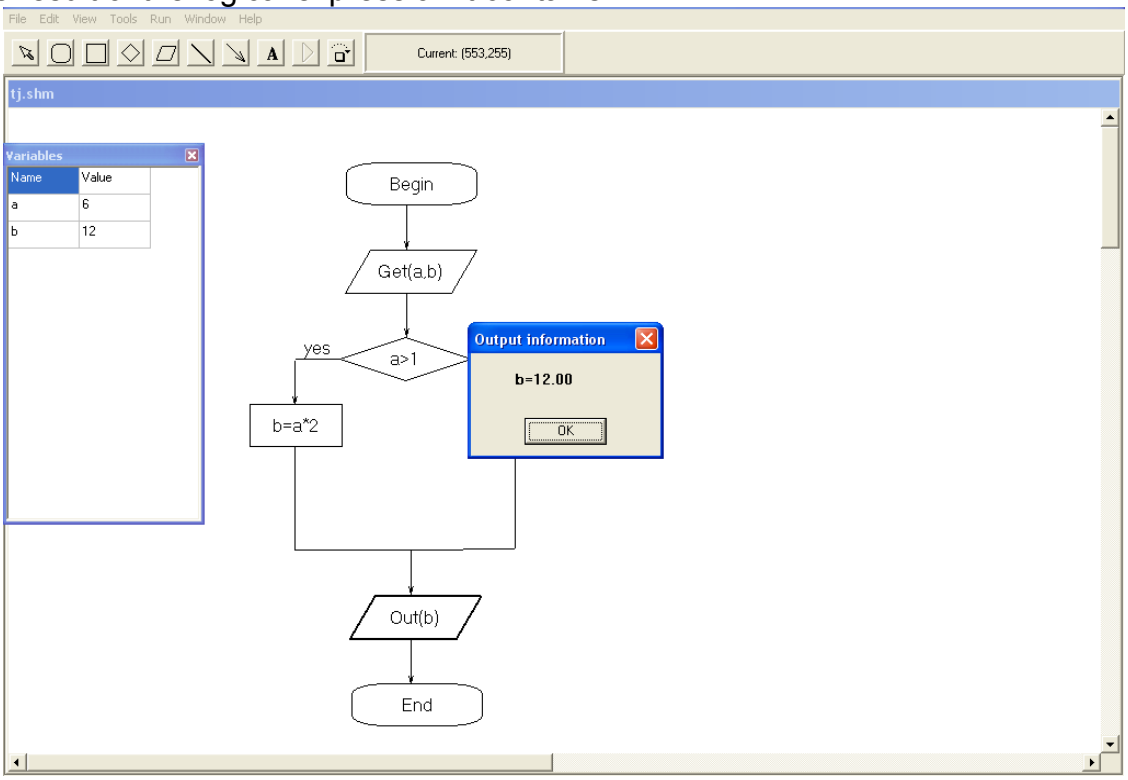

Figure 3. Variable Window and Output Result

Conclusions and future work

Future plans are to combine the use of the tool with a well-defined teaching pedagogy and online learning environment. Currently, the used pedagogy is based on operational approach theories of learning development [15] and it will being investigate. Of particular interest are the solving problems principles of this classroom teaching pedagogy. However, this idea is still in the initial stages of development and much more work is needed. Firstly, further research is needed to refine the ideas regarding this teaching pedagogy and how it can best be used to aid in the delivery of the introductory concepts of programming. Secondly, an online learning environment that fully exploits the capabilities of our algorithm animation tool and the pedagogy needs to be researched, designed and

built. Modification of the tool will also be needed to further its use with respect to these proposals.

REFERENCES

[1] Byrne, P. & Lyons, G., The effect of student attributes on success in programming, Proceedings of the 6th annual conference on Innovation and technology in computer science education, 2001, pp. 49-52.

[2] Kölling, M. & Rosenberg, J., Blue - A Language for Teaching Object-Oriented Programming, Proc. of the 27th SIGCSE Technical Symposium on Computer Science Education, 1996, pp. 190-194

[3] Kurland, D., Pea, R., Clement, C. & Mawby, R., A Study of the development of programming ability and thinking skills in high school students. In Soloway & Spohrer: Studying the Novice Programmer, 1989, pp. 209-228.

[4] London Metropolitan University. Learning Objects for Introductory Programming. http://www.unl.ac.uk/ltri/learningobjects/index.htm, referenced 2.12.2002.

[5] McCracken, M. Almstrum, D. Diaz, M. Guzdial, D.Hagen, Y. Kolikant, C. Laxer, L. Thomas, I. Utting and T. Wilusz, , A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-year CS Students, SIGCSE Bulletin – Volume 33 – Issue 4, ACM Press, NewYork – NY –US, 2002, pp 125-140

[6] Naps, T., G. Roessling, et. al., Exploring the Role of Visualization and Engagement in Computer Science Education, report of the working group on Visualization at the ITiCSE conference in Arhus, Denmark, 2002.

[7] Naps, T. L, G. Roessling, J. Anderson, S. Cooper,W. Dann, R. Fleischer, B. Koldehofe, A. Korhonen, M. Kuittinen, C. Leska, L. Malmi, M. McNally, J. Rantakokko, R. Ross, Evaluating the Educational Impact of Visualization, inroads - Paving the Way Towards Excellence in Computing Education, volume 35, Number 4., 2003, pp. 124-136, ACM Press, New York.

[8] Pane, J. & Myers, B. Usability Issues in the Design of Novice Programming Systems, School of Computer Science Technical Reports, Carnegie Mellon university, CMU-CS-96-132, 1996, available at http://www.cs.cmu.edu/~pane/ftp/CMU-CS-96-132.pdf

[9] Perkins, D., Hanconck, C., Hobbs, R., Martin, F. & Simmons, R., Conditions of learning in novice programmers. In Soloway & Spohrer: Studying the Novice Programmer, 1989, pp. 261-279.

[10] Roberts E, An Overview of Mini Java, Proceedings of the 32nd SIGCSE technical symposium on Computer Science Education, Volume 33 Issue 1, ACM Press, New York - USA, 2001, pp: 1-5.

[11] Robins A, Rountree J, and Rountree N, Learning and teaching programming: A review and discussion, Computer Science Education - Volume 13 Issue 2, Routledge, Oxford England, 2003, pp: 137-172

[12] Smith P and Webb G, An Overview of a low-level Program Visualisation Tool for Novice C Programmers, In Proceedings of the Sixth International Conference on Computers in Education (ICCE '98) - Volume 2, Springer-Verlag, Beijing, China, 1998, pp: 213-216.

[13] Soloway, E. & Spohrer, J. Studying the Novice Programmer, Lawrence Erlbaum Associates, Hillsdale, New Jersey. 1989, p. 497

[14] Winslow L., Programming Pedagogy -A Psychological Overview. SIGCSE Bulletin – Volume 28 Issue 3, ACM Press, New York USA, 1996, pp. 17-22.

[15] Леонтьев, А. Н. Деятельность, сознание, личность. Моссква, 1975

CONTACT ADDRESSES
Pr. Assist. Galina Atanasova
Department of Natural Science and Education,
Angel Kanchev University of Ruse
Phone: (+359 82) 888 326
E-mail: gea@ami.ru.acad.bg

Assos. Prof. Katalina Grigorova,
Department of Natural Science and Education,
Angel Kanchev University of Ruse,
Phone: (+359 82) 888 464,
E-mail: katya@ami.uni-ruse.bg.

СРЕДСТВО ЗА ОБУЧЕНИЕ НА НАЧИНАЕЩИ ПРОГРАМИСТИ

Галина Атанасова, Каталина Григорова

Русенски университет „Ангел Кънчев”

Резюме: *Статията разглежда особеностите на начинаещите програмисти, техните затруднения във въвеждащите курсове по програмиране. Обърнато е внимание на многоаспектността на материала и причините, поради които начинаещите го намират за труден, непреодолим и се демотивират. Представено е положителното въздействие на средите за визуализация и анимация на алгоритми. Статията подчертава важността начинаещите да усвоят умения за съставяне на коректни и ефективни алгоритми, за да постигнат успех в програмирането. Представено е средство за подпомагане на начинаещите програмисти да преодолеят трудностите и да изградят умения за съставяне на алгоритми.*

Ключови думи: *Компютърни науки, Алгоритми, Начинаещи програмисти, Анимирано представяне на алгоритми*

ISSN 1311-9974