Information Horizons: AMERICAN Journal of Library and Information Science Innovation
Volume 02, Issue 05, 2024 ISSN (E): 2993-2777

Innovative Methods and Resources for Programming Education

Najmiddinova Khilola Yokubjanovna

doctor of Pedagogical Sciences, professor, Namangan State University

Abstract. *There are certain unique challenges when teaching programming at the university beginning level, such as the vast range of student's prior knowledge, fear of programming, issues with syntax in programming languages, etc., as demonstrated by several earlier studies in our online survey. Several methods and resources have already been created to help students learn programming ideas more easily. These include various visual aids, guidance, video lectures, and even brand-new programming languages. Rather than using ordinary IDEs, our method is centered on the development of specialized learning interfaces for common programming languages like C++. It should be easier for professors to assist their students before they accumulate a lot of syntax and logical errors thanks to this interface, which should also prevent many of the common mistakes that students make when learning programming.*

Keywords: *programming, teaching software, teaching approaches*

Introduction

This study examines the unique characteristics and challenges associated with teaching programming to university beginners. It proposes our suggestions for structuring the teaching process. The process refers to the systematic steps involved in acquiring programming knowledge and abilities. The software tool supports this process and facilitates students in achieving the required programming knowledge and skills more easily.

Over the past few years, we have regularly administered a web-based questionnaire to a sample of students enrolled in the first year of study at the Faculty of Organization and Informatics, namely in the Programming 1 course. The questions pertained to the student's prior programming experience and overall knowledge of informatics, as well as any specific challenges they encountered in comprehending programming. Their responses aided us in identifying their challenges and obstacles in the learning process, as well as our main concerns in the teaching approach. Firstly, it is evident that teaching programming is confronted with a wide range of student's prior knowledge, as well as their diverse attitudes toward programming. The significant challenge in teaching programming lies in its distinctiveness compared to other disciplines, as the amount of prior knowledge varies significantly. Moreover, several pupils have apprehension towards programming, perceiving it as an exceedingly challenging task. Many students initially adopt a "it's easy" mindset, but as the exercises grow more challenging, they quickly shift to a "it's too difficult for me" perspective. Typically, we draw a parallel between that scenario and the exercise room when new trainees join. They frequently display great enthusiasm towards their training but occasionally fail to take warnings regarding their practice seriously, such as starting with excessively large weights. Consequences, such as pain, bruises, and even hernia, may occur as students strive to acquire programming abilities, and these consequences also have corresponding mental implications. There are various methods to address such issues, and

we mentioned some of them in our chapter on Related work.

Our technique involves creating a specific learning interface for mainstream programming languages, such as C++, rather than using ordinary Integrated Development Environments (IDEs). The learning interface should incorporate measures to discourage students from engaging in poor programming habits, such as producing code without proper syntax and logical validation, as well as relying solely on memorization to learn program code. There are additional benefits for teachers and the teaching process, such as the ability to more easily assist students before they make mistakes and prevent them from engaging in unacceptable behavior, such as copying programs (programs must be written during exercises).

There are numerous inquiries concerning the instruction of computer programming, specifically regarding teaching approaches and the technology employed. While it is ideal for a good programming course to be independent of the programming language used, it is beneficial to teach the fundamentals of computer programming about a specific programming language. The selection of a programming language is of utmost importance, if not essential, for the structuring of a programming course. As evidenced by reference [34], modern programming languages are increasingly gravitating towards C-like programming languages. Over 50% of computer code utilized in the United States is composed in one of the languages closely related to the C programming language, namely C, C++, C#, and Java. PHP and Visual Basic are the only two other programming languages that are utilized considerably. PHP accounts for over 10% of the used code, while Visual Basic accounts for 8.5%. There are several criteria we considered when selecting a programming language:

- Utilization of programming languages. Support for widely used operating system platforms
- Comprehensive coverage of essential programming principles.

The initial parameter in the list unambiguously refers to one of the programming languages that resemble C. Despite being the most widely used programming language now and having the advantage of being platform agnostic, Java's suitability as a choice is called into doubt due to the omission of crucial concepts like pointers. It would not pose a significant challenge for students whose primary area of study is not computer software design and development to take the programming course. Programming courses for computer and information science students must cover all essential programming principles, including pointers and dynamic memory allocation algorithms. The C programming language is commonly utilized for introductory programming courses, followed by more complex languages such as C++ or Java. In our perspective, commencing with the C programming language is deemed unnecessary. Instead, we believe that beginners in computer programming would find it more advantageous to begin with other C-like languages that offer a greater degree of programming. Hence, we choose to employ a solitary programming language across all degrees of programming training. C# is the most recent among the four most widely used programming languages that are similar to C. The programming language offers comprehensive support for all fundamental programming ideas, including intriguing concepts that are particularly useful for educational purposes, such as secure pointers and the option for manual or automatic memory reallocation. The disadvantages of C# include its relatively low adoption rate (about 4.3%) and limited support for the Linux platform.

Conclusion.

Teaching programming at the beginner's level in university encounters certain special challenges, as indicated by the questionnaire completed by our students. The absence of a prior. The acquisition of knowledge, along with difficulties in comprehending program code and sometimes apprehension towards programming, can often steer pupils in the incorrect direction. Poor programming habits, such as writing code without proper syntax and logical testing, as well as relying on rote

memorization to learn program code, can result in significant issues when tackling more challenging exercises. These habits often lead students to the conclusion that programming is "too difficult" for them. Our strategy for addressing this issue involves acquiring proficiency in programming interfaces for standard compilers. This will discourage students from copying programs from one another and instead encourage them to incorporate checkpoints throughout the development of their programs. Additionally, there are other options available to assist students in the analysis and debugging of software code. Every program created using the learning programming interface is customized by including the data submitted in a form at the start of the programming session, together with additional data and an MD5 checksum. This checksum ensures that the program was indeed produced using the learning programming interface. Using a learning programming interface offers benefits for both students and the teaching process. Students are encouraged to thoroughly test their programs during the development phase to prevent any problems from occurring. In addition, the software analysis and debugging tools assist them in identifying the root cause of their errors. In the educational process, it is crucial to prevent students from plagiarizing their programs. Furthermore, teachers can assist their pupils in rectifying their syntax and other faults before they accumulate a significant amount. In our future development of the programming interface, we aim to incorporate several enhancements. These include improved explanations of syntax and logical errors, the inclusion of time limitations for exams, and the addition of program-related questions to assess the student's comprehension.

References:

1. Abernethy, K., Piegari, G., Reichgelt, H., Treu, K.: An Implementation Model for a Learning Object Repository, Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, Ostober 24-28, Vancouver, Canada, 2005, pp. 2- 7.
2. Najmiddinova, X. Y., & Toxirjonova, X. Y. (2022). Some Examples of Automorphism in A Limited Group. *International Journal on Integrated Education*, *5*(6), 497-500
3. Begel, A.: LogoBlocks: A Graphical Programming Language for Interacting with the World, Boston, MA, MIT, 1996.
4. Najmiddinova, K. Y. (2020). DETERMINATION OF THE LEVEL OF MATHEMATICAL LITERACY USING COMPUTER GAMES. *Scientific Bulletin of Namangan State University*, *2*(1), 413-419.
5. Brusilovsky, P. L.: Turingal - the language for teaching the principles of programming, Proceedings of Third European Logo Conference, 27-30 August, Parma, Italy, 1991, pp. 423-432.
6. Najmiddinova, K. Y. (2021, January). INFLUENCE OF FAMILY ON THE DEVELOPMENT OF MATHEMATICAL LITERACY OF CHILDREN. In *Archive of Conferences* (Vol. 13, No. 1, pp. 120-128).
7. Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., Miller, P.: Mini-languages: A Way to Learn Programming Principles, Education and Information Technologies, Vol. 2, No. 1, pp. 65-83.
8. Нажмиддинова, Х. (2023). О ПРОБЛЕМАХ ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОГО ОБУЧЕНИЯ СТУДЕНТОВ В ОБУЧЕНИИ НА ОСНОВЕ КРЕДИТНО-МОДУЛЬНОЙ СИСТЕМЫ. *Namangan davlat universiteti Ilmiy axborotnomasi*, (7), 776-782.
9. Viner, N. Kibernetika, ili Upravleniye i svyaz' v zhivotnom i mashine [Tekst] / N. Viner ; per. s

angl. I. V. Solov'yeva i G. N. Povarova ; pod red. G. N. Povarova. – 2-ye izdaniye. – M. : Nauka ; Glavnaya redaktsiya izdaniy dlya zarubezhny'h stran, 1983. – 344 s.

10. Duvanov, A. A. Azbuka Robotlandii. Informatsiya [Tekst] / A. A. Duvanov [i dr.] // Informatika, ID Pervoye sentyabrya. – 2012. – № 7. – S. 36–49, elektronnoye prilozheniye na CD.
11. Yeremin, Ye. A. Sreda Scratch – pervoye znakomstvo. «Informatika» [Tekst] / Ye. A. Yeremin // IMD 1 sentyabrya. – 2008. – № 18 – S. 17–24, № 20 – S. 16–28.
12. Yershov, A. P. Shkol'naya informatika (kontseptsii, sostoyaniye, perspektivy) [Tekst] / A. P. Yershov, G. A. Zvenigorodskiy, Yu. A. Pervin. – Preprint VTS SO AN SSSR, №152, Novosibirsk, 1979. – 26 s.
13. Zvenigorodskiy, G. A. Osnovny'ye operatory' uchebno-proizvodstvennogo yazy'ka Rapira [Tekst] / G. A. Zvenigorodskiy // Kvant. – 1980. – № 1. – S. 52–55.