Mathematical and Software Engineering, Vol. 11, No. 1-2 (2025), 1-4
Varɛpsilon Ltd, http://varepsilon.com
DOI 10.5281/zenodo.17690957

The role of thinking in programming

Teodosi Teodosiev

Faculty of Mathematics and Informatics, University of Shumen, Bulgaria

Abstract

This paper explores the bidirectional relationship between thinking and programming. It discusses the role of thinking in programming education and examines whether programming can foster higher-order thinking skills. Attention is given to certain challenges in programming education that conflict with the principles of constructive thinking.

Keywords: thinking, learning, programming, AI, ChatGPT.

1 Introduction

The inspiration for this work came from my desire to test ChatGPT with a problem originally published about 30 years ago in *PC Magazine Bulgaria* by my teacher and colleague, Assoc. Prof. Dimitar Dobrev (in memoriam) [1]

The problem is simple: enter two integers and display the relationship between them (>, =, or <). To make the task more interesting, several strict constraints are imposed on the control structures allowed. Six versions were considered: using only selection statements, only loops, arrays and loops, arrays and assignment, arrays only, or a single algebraic expression.

AI (ChatGPT) managed to solve the problem but required guiding questions. This led me to a key reflection: must we develop logical and constructive thinking to fully leverage AI's potential and to evaluate the correctness of its answers?

With the growing influence of AI, many young people tend to rely on it instead of engaging in independent reasoning — after all, AI seems able to answer any question.

2 Examples

Motivated by the previous task, I decided to test AI on two simple problems that I have long used in my *Introduction to Programming* course to demonstrate to students the importance of careful algorithmic reasoning.

The first task is to compute the arithmetic mean of all integers in the interval [a, b]. When presented to students, about 90% (based on my observations) immediately begin coding based solely on the definition of arithmetic mean. The implementation requires a loop to calculate the sum and count of the numbers. This task illustrates Dijkstra's well-known idea that "...my main concern in teaching computing science was to train students to think first and not to rush into coding..." [2]. Beginners must spend sufficient time reasoning on paper before sitting at a computer.

BgGPT's initial solution used a loop to compute the sum — the standard approach. When prompted for an alternative, it produced a solution based on the arithmetic progression formula, and after further prompting, the simplest but less common version: (a + b) / 2.0.

The second task is to evaluate a polynomial at a given point, provided its coefficients. Similar patterns emerged: BgGPT first produced a standard loop-based C++ solution using pow(x, i) with a vector of coefficients, followed by array and dynamic-memory versions. Only when I suggested using Horner's method did it generate the optimal and most efficient solution.

With well-structured and targeted prompts, we can guide AI toward optimal solutions — but to do so, we ourselves must possess sound knowledge and critical thinking skills.

3 Thinking

In daily life, technology increasingly reduces the need for us to think or know things independently. Yet in professional practice, knowledge and skills remain essential. "Learning to think can be considered the main objective of education" [3], and this is especially true in programming.

Programming and thinking are interdependent processes. Education should teach students how to think critically and constructively — even when using AI. Logical and constructive thinking are fundamental in programming education.

In the learning process, solving a problem is not an end in itself but a means toward developing problem-solving strategies. The process is more important than the final answer. Learning programming requires mental effort, concentration, logic, and imagination. Algorithmic thinking is a key skill for every programmer.

Programming demands a particular *style of thinking* — the ability to represent algorithms in a programming language clearly and effectively. Many difficulties stem from unclear task modeling or an inability to consider all relevant factors — in short, from unclear thinking. Therefore, the development of clear and structured thinking must be a central goal in programming education.

Challenging, constraint-based tasks encourage creative problem-solving and nonstandard reasoning. An inverse relationship can be observed between the level of cognitive effort and the level of programming proficiency — higher levels of thinking can compensate for less technical experience. Introductory programming should thus emphasize algorithmic reasoning, which relies on logic and mathematical discipline.

Some researchers argue the opposite — that programming itself enhances thinking. As stated in [3], both programming and constructive thinking involve at least four core skills: (a) problem decomposition; (b) pattern recognition; (c) abstraction; and (d) algorithmic reasoning.
Steve Jobs' famous quote — "Everyone should learn to program because it teaches you how to think" — further supports this link. Programming education can foster constructive and higher-order thinking. Constructive thinking is a crucial skill for the 21st century, and programming provides a natural means of developing it by engaging learners in abstraction, sequencing, parallelism, and debugging [4].

4 Programming Education

As Wirth [5] wrote: "Considering our own subject, the field of computing and programming, an academic institution's ultimate goal must be much wider than the mastery of a language. It must be nothing less than the art of designing artifacts to solve intricate problems. Some call it the art of constructive thinking."
Another important aspect is assessment. In the past two decades, testing has become the dominant method of evaluating student learning. Multiple-choice tests are often praised for their standardization, objectivity, automation, and broad coverage. However, they are not without drawbacks — particularly in disciplines like mathematics and informatics, which are inherently constructive. Such tests often measure guessing ability rather than true understanding.
According to Wirth's vision, and based on other challenges such as modeling, algorithm design, and general academic preparation, multiple-choice testing is not suitable for assessing constructive thinking. These tests may verify syntax and semantics (formal aspects) but fail to evaluate algorithmic design and creative reasoning (the essence of programming).
In recent years, many instructors have relied on ready-made code examples, often discussing them directly in class. While this can improve efficiency and reduce errors, it risks concealing the *thought process* behind program design. When code is presented without showing the reasoning steps, students lose the opportunity to develop their own analytical thinking.

5 Conclusion

When smartphones became widespread about thirty years ago, they were quickly adopted by young people — often replacing basic mental arithmetic with calculators. As a result, many students today struggle to perform simple calculations mentally. A similar risk may accompany the rise of AI: increasing dependence on technology at the expense of independent thought.
With AI's rapid development, more and more young people turn to it for answers instead of thinking for themselves. Yet, as Dijkstra reminded us decades ago:
Think first! [2]

Acknowledgements
This work is partially supported by the Scientific Fund "Scientific Research" at the University of Shumen, Bulgaria – Contract № RD-08-75-31.01.2025.

References

[1] Dobrev, D., Concerti piccoli virtuosi per computer virussioso, PC Magazine Bulgaria, April, 1996, 104-107.
[2] Dijkstra E, “Why is software so expensive?” An explanation to the hardware designer,1982, https://www.cs.utexas.edu/~EWD/ewd06xx/EWD648.PDF.
[3] Hermida, J. (2024). Is learning computer programming associated with developing thinking skills? UNESCO International Bureau of Education – IBE Blog https://www.ibe.unesco.org
[4] Belmar, H., Romero, M., & de Benito, B. (2022). *Review on the teaching of programming and computational thinking*. Frontiers in Computer Science, 4, 883377. https://doi.org/10.3389/fcomp.2022.883377
[5] Wirth N, Computing Science Education: The Road not Taken, ITiCSE Conference, Aarhus, Denmark, 2002, http://www.softwareschule.ch/download/wirthcomputing_education.pdf