Eximia Journal

(ISSN 2784-0735)

Vol. 13

2024

Eximia Journal
Vol. 13, 583-589, January, 2024
ISSN: 2784-0735
www.eximiajournal.com

Programming Teaching in the Era of Artificial Intelligence

Chien-Hsing Huang

Department of Information Engineering, I-Shou University, Taiwan, R.O.C.

raylan@isu.edu.tw

Abstract. With the development of science and technology, computer applications are changing with each passing day, changing our lives in all aspects. Computers have become an indispensable tool in life, and learning programming languages to operate computers has become a major focus of education. From low-level languages to high-level languages to visual programming languages, and finally to generative artificial intelligence to generate code. Program development tools are constantly updated as computers advance. This has the benefit of lowering barriers to learning, but also results in a weaker understanding of how the program works. This article designs a programming language learning strategy that combines the above programming development tools to lower the threshold for programming language learning and improve learning efficiency.

Keywords. Programming Teaching, Visual Programming Languages, Generative Artificial Intelligence

1. Instruction

In this era of booming information development, cultivating information technology literacy has become a new trend in education around the world. The United States, the United Kingdom, Germany, the Netherlands, Japan, mainland China, Australia, Israel and other countries have popularized and opened information courses in education to strengthen the cultivation of key talents such as computational thinking, information technology, and practical abilities.

Computational thinking is about identifying the basic computing capabilities and limitations of equipment and using these computing capabilities to solve problems. Different devices will have different computing capabilities and limitations. Incorrect use of calculations can lead to errors of principle. For example, errors are impossible in human arithmetic operations. The representation range is limited, and the data representation method in computer equipment also limits the representable range.

Research points to a correlation between computational thinking skills and programming activities. Learning through procedures helps to cultivate the connotation of computational thinking, such as decomposition, generalization, etc. In order to simplify the learning process of computational thinking, people have developed visual programming tools, such as scratch, m-block, Greenfoot, Alice, code.org and blockly, etc., to help programming beginners establish strategies to solve problems.

Eximia Journal
Vol. 13, 583-589, January, 2024
ISSN: 2784-0735
www.eximiajournal.com

Visual programming can improve the learning motivation of programming beginners and help them become familiar with the syntax and structure of programming. But it cannot be put into actual use. Just playing games, not really writing programs. Real programming learning still requires returning to programming language. How to enable learners to overcome the obstacles from graphic puzzles to text encoding is the key to programming teaching. The first difficulty faced.

With the rise of artificial intelligence, generative artificial intelligence can be used to understand human thinking and convert it into program code for device computing thinking. It has become a shortcut to program development. However, the provided code often fails due to insufficient information provided by the user. A fallacy has occurred. If you want to be able to provide enough information, you must have good programming skills, so that programming learning can return to the starting point.

This article proposes a strategy for using visual programming tools and generative artificial intelligence to assist in learning programming concepts and implementing programming codes: visual programming cultivates programming concepts, and generative artificial intelligence translates human thinking and computational thinking to assist in generating program code, develop the ability to design programs to solve real-world problems.

2. Visual programming language

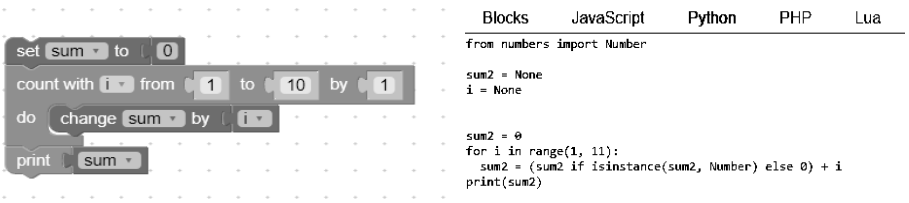
Visual Programming Language (VPL), also known as Graphical Programming Language (GPL), is a programming language that allows users to use graphic elements for programming. Easier than text programming. VPL is based on visual expression, using grammar or some kind of auxiliary markup to arrange graphics and text. Many VPLs are based on the concepts of blocks and arrows, with blocks or objects on the screen as the main body, connected by arrows, and straight-line segments and arc segments representing the relationship between them.

Visual programming languages are simple tools for implementing computational thinking. The graphics blocks provided by the programming language represent the basic computing functions of the programming language. Graphical blocks are combined to construct command sequences. Lines or arrows guide the flow of data or the order of execution. Although the vision of natural visual programming cannot be easily converted into written code, games can help users learn to use computational thinking to solve problems, which can be used to verify the running process of the program.

The average person's mind does not consider the details of operations. For example, consider turning on the air conditioner when the weather is hot. In computational thinking, heat must first be quantified through temperature, and an appropriate critical value must be set as a condition for judging heat sensation. Finally, logical operations can be used to determine whether air conditioning should be used. Beginners in programming must first learn to use VPL to transform human thinking into computational thinking. This is the first step in learning programming.

Figure 1 shows an example of conversion between visual programming language and text coding programming language: using Blockly puzzle to calculate the sum of 1 to 10, and then converting this puzzle into python language.

Eximia Journal
Vol. 13, 583-589, January, 2024
ISSN: 2784-0735
www.eximiajournal.com



(a)Blockly Puzzle (b) Python Codes

Fig1. Example of conversion between visual programming language and text encoding programming language

3. Integrated Development Environment

If you want to complete a job correctly, you must first understand what tools are available at hand. Only by using the tools properly can you solve the problem correctly. Integrated Development Environment (IDE), also known as Integration Design Environment or Integration Debugging Environment, is an application software that assists program developers in developing software, integrating editing, construction, testing and packaging functions.

Use the declarative rules of programming language to train artificial intelligence, so that it has the ability to automatically write or edit original code, give specific font colors to keywords in the programming language, visual cues make the original code more readable, and program code writing Instant feedback can also be provided if grammatical errors occur accidentally.

Making good use of an IDE can make programming as easy as a visual programming language. However, the loss of gameplay will reduce interest in learning, but the sense of accomplishment in completing the code can also stimulate learning motivation. There is no need to spell out the entire code keyword, just enter the first letter of the keyword and a prompt will appear. No need to remember the complete command syntax. After selecting the required keywords, the relevant grammar will dynamically prompt the learner to complete it, allowing the algorithm of computational thinking construction to be quickly converted into program code.

4. Programming Language

Each programming language is designed considering factors such as actual needs, execution environment, core programming ideas... and has different functions, syntax, and architecture. Understanding the core design of programming languages will provide a deep understanding of computational thinking and program implementation. Big help, generally speaking, programming languages can be divided into the following categories:

- Procedural Programming Languages: Procedural languages follow a sequence of statements or instructions to achieve the desired output. Each series of steps is called a process, and a program written in one of these languages will contain one or more programs within it. Common programming languages such as C, C++, Java, Pascal, BASIC, etc.
- Functional programming languages: Reusable modules of program code are called functions. Functional languages do not focus on the execution of statements, but on the output of mathematical functions and evaluations. Each function performs a specific task and returns the results. Results will vary based on the information you enter into the function. Common functional programming languages include Scala, Erlang, Haskell, Elixir, and F#.
- Object-oriented programming (OOP): This type of language treats a program

Eximia Journal
Vol. 13, 583-589, January, 2024
ISSN: 2784-0735
www.eximiajournal.com

as a set of objects consisting of data and program elements (called properties and methods). Objects can be reused in one program or other programs. This makes it a popular language type for complex programs because the code is easier to reuse and extend. Common object-oriented languages such as Java, Python, PHP, C++ and Ruby.

- Scripting languages: Programmers use scripting languages to automate repetitive tasks, manage dynamic web content, or support processes in large applications. Common scripting languages include: PHP, Ruby, Python, bash, Perl, and Node.js.
- Logic programming language: Logic programming language does not tell the computer what to do, but expresses a series of facts and rules to guide the computer how to make decisions. Some examples of logic languages include: Prolog, Absys, Datalog, and Alma-0.

Learning to convert operational thinking into the core thinking of programming language will not only help you write program codes, but also help you understand the principles of program operation. For example, the for loop in C language is planned in the form of a sequence. The focus of learning is on what is needed for design. Sequence; the for loop in python is planned in the form of a set, and the focus of learning is to design the elements in the set. The visual programming language can be set to be executed repeatedly several times, which belongs to computational thinking, and then needs to be converted into procedural thinking to complete the program coding.

5. Programming by generating artificial intelligence

Artificial intelligence can be defined as an adaptive automation program that automatically adjusts the content of the automation program based on demand parameters input by the user. What can be automated depends on the scope covered by the training set. Many research institutions and large enterprises have invested in artificial intelligence research and development and made the results public. Free or paid artificial intelligence has penetrated into daily life and is within reach.

Generative AI is a type of artificial intelligence that can create new content and ideas, including creating conversations, stories, images, videos, and music. Artificial intelligence technology attempts to imitate human intelligence in non-traditional computing tasks, such as image recognition, natural language processing(NLP) and translation. Programming, a job that follows logical rules, is more suitable for training artificial intelligence to learn programming languages. Generative AI reuses training data to generate problem-solving code. As long as the user can clearly describe the problem, generative AI can produce coded prototypes that solve the problem.

Current generative artificial intelligence already has the ability to transform human thinking into computational thinking. It can even generate program code using trained programming languages such as Python or C++, and can be used as a personal assistant for programming teaching. The learning process becomes a trial-and-error process, with learners learning by manipulating generative AI responses. The learning process will not be interrupted by setbacks, and the enthusiasm for learning will continue.

Eximia Journal
Vol. 13, 583-589, January, 2024
ISSN: 2784-0735
www.eximiajournal.com

Figure 2 shows the use of generative artificial intelligence to generate C++ code to calculate the sum of 1 to 10.

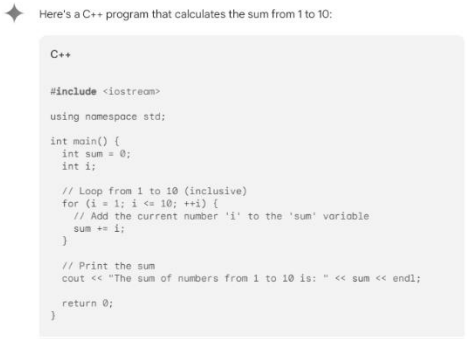


Fig2. Using generative artificial intelligence to generate C++ code

6. Online Judge System

Checking code for correctness is a difficult task. The compiler can check whether there are syntax or logic errors, but verifying whether the program code can correctly solve the problem and meet the requirements of the problem is still an unsolvable problem, so an online judgment system is generally used to handle this work. Although the verification results are not guaranteed to be correct, they can provide a relatively fair assessment.

Online Judge (OJ) is an online system for testing algorithmic competition entries. This is for general program practice. The principle is black box testing. Exam materials are prepared in advance according to the question requirements. Test data consists of paired data of inputs and corresponding outputs. The user submits the source code to the server. The server compiles the user's source code, then executes the executable application generated by the source code, obtains the output, and compares it with the correct execution result. Scoring is based on the number of test questions a user-submitted program can answer correctly, the number of attempts the user makes, and the execution time of the submitted code.

OJ allows learners to immediately known whether the code they wrote is correct. At the same time, it can detect errors in the program code based on feedback from test data. For example, programmers fail to consider the data storage type and treat floating-point data as integer data. Although the program idea is correct, the output result of program execution is wrong.

7. Programming Learning Design

Programming learning planning is divided into six parts: learning planner, visual programming language, programming language development interface, generating artificial intelligence, online assessment system and programmable learner. Learning planners use a problem-based approach to setting learning goals. , programming learners use visual programming languages, programming language development interfaces, and generative artificial intelligence to attempt to write code that can be tested through an online review system.

The Learning Pyramid shows the average learning retention rate of learners after two weeks of using different learning methods. Among them, the most effective learning is immediate application and practical operation. In this learning program, learners must use tools to solve problems posed by the planner. During the operation, artificial intelligence plays the interactive role of teaching assistant, peers and students at the same time, which can achieve the highest learning efficiency. At the same time, learning emotions and learning environment are not considered in the learning pyramid. The timely feedback mechanism can maintain the

enthusiasm for learning, and the artificial intelligence interaction mode will also be dynamically adjusted to the best according to the user.

8. Conclusions

In the early days of their development, programming languages were the tools of scientists and engineers. As the technology matures, it becomes an essential skill in technology and engineering. With the popularity of computers, it has become an important subject in higher education. Now as information technology has penetrated into daily life, the learning of programming languages has also extended to primary and secondary education.

Generally speaking, programming language courses in primary education teach visual programming language and allow students to learn computational thinking; programming language courses in secondary education teach functional programming language and use function calls to replace graphical blocks in visual programming language; Programming language courses in higher education teach core programming language techniques such as data structures, algorithms, and compilers. Although it is step-by-step, it is easy for cognitive gaps to occur in the connection of courses, causing students to have conflicts such as "Why use troublesome tools to do the same work?"

In the study plan of this course, each unit sets learning goals according to the problem, appropriately adjusts the proportion of use of various tools in the course, and can be connected and integrated with programming language courses in primary education, secondary education and higher education. It can even evolve into a form of lifelong self-learning.

References:

[1] Al-Imamy, S., Alizadeh, J., & Nour, M. A. (2006). On the development of a programming teaching tool: The effect of teaching by templates on the learning process. Journal of Information Technology Education: Research, 5(1), 271-283.

[2] Boshernitsan, M., & Downes, M. S. (2004). Visual programming languages: A survey. Los Angeles, CA, USA: Computer Science Division, University of California.

[3] Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada, 1–25.

[4] Burnett, M. M., & McIntyre, D. W. (1995). Visual programming. COmputer-Los Alamitos-, 28, 14-14.

[5] Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. Computers & Education, 95, 202–215.

[6] Davis, B., & Summers, M. (2014). Applying Dale's Cone of Experience to increase learning and retention: A study of student learning in a foundational leadership course. Qscience proceedings, 2015(4), 6.

[7] Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational researcher, 42(1), 38-43.

[8] Johnson, L. F. (1995). C in the first course considered harmful. Communications of the ACM, 38(5), 99-101.

[9] Kanika, Chakraverty, S., & Chakraborty, P. (2020). Tools and techniques for teaching computer programming: A review. Journal of Educational Technology Systems, 49(2), 170-198.

[10] Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. ACM SIGCSE Bulletin, 39(1), 223-227.
[11] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(11), 60-67.
[12] Salleh, S. M., Shukur, Z., & Judi, H. M. (2013). Analysis of research in programming teaching tools: An initial review. Procedia-Social and Behavioral Sciences, 103, 127-135.
[13] Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition.