New Trends and Issues Proceedings on Humanities and Social Sciences

Volume 8, Issue 2, (2021) 11-19

www.prosoc.eu

Selected paper of Selected papers of 11th World Conference on Educational Technology Researches (WCETR-2021) 02-03 June 2021
University of Kyrenia, Kyrenia, Cyprus
(ONLINE VIRTUAL CONFERENCE)

Programming design, education, innovation, exploration and practice based on students' cognitive laws

Lin Wan*, School of Software, Huazhong University of Science and Technology, Wuhan 430074, China

He Tang, School of Software, Huazhong University of Science and Technology, Wuhan 430074, China

Shaohong Fang, School of Software, Huazhong University of Science and Technology, Wuhan 430074, China

Suggested Citation:

Wan, L., Tang, H. & Fang, S. (2021). Programming design, education, innovation, exploration, and practice based on students' cognitive laws. *New Trends and Issues Proceedings on Humanities and Social Sciences*. *8*(2), 11–19. Available from: www.prosoc.eu

Received from June 20, 2021; revised from July 25, 2021; accepted from September 05, 2021.
Selection and peer review under responsibility of Prof. Dr. Huseyin Uzunboylu, Higher Education Planning, Supervision, Accreditation and Coordination Board, Cyprus.

Abstract

This paper analyses the cognitive characteristics and cognitive laws of freshmen students in learning the C programming language. We have built a spirally ascending complete curriculum theory knowledge system and a practical teaching model of *experience–verification–exploration.* The knowledge system is established through programming thinking from the perspective of computer systems, designed according to the teaching structure of knowledge construction and trained based on the engineering literacy with a new engineering background. The implementation of the teaching reform has achieved the expected results. The curriculum theory knowledge system is complete, and the practical teaching content is novel and rich. It has stimulated students' learning enthusiasm and innovative consciousness, cultivated students' professional qualities and helped to cultivate talents that meet the needs of the industry.

Keywords: Programming language, perspective, computer systems, knowledge construction.

* ADDRESS FOR CORRESPONDENCE: **Lin Wan,** School of Software, Huazhong University of Science and Technology, Wuhan 430074, China.
E-mail address: hetang@hust.edu.cn

1. Introduction

The group of IEEE-CS and ACM announced the Computer Science Curricula 2013. It claimed that having a *system level vision, open subject vision* and *being a full-scale professional* are new requirements for computer-majoring students in the era of network economy (Curricula, 2001; Qin & Tan, 2016). For computer majors, the *C programming language* course for freshmen is the first professional basic course to learn; it is not only necessary to help students cultivate programme design ideas and master programme design methods on the basis of establishing the concept of *system level*, but also to help students expand their subject horizons and improve engineering literacy.

In recent years, some scholars have explored the cognitive rules of learners in programming language learning. Xinogalos, Pitner, Ivanovic and Savic (2018) concluded that there is no significant difference between C-like language and Pascal-like language in learning difficulty for beginners. Teaching reform of Chen and Zhou (2017) is based on the integration of C programming language and data structure for the cultivation of practical programming skills. Based on the method of cognitive task analysis, Zhao and Chen (2020) expounded how teachers can help programmers to acquire expert knowledge needed for analysis and problem solving, providing a new way for teachers to carry out programming teaching. For freshmen, C language programming is their entry language. This paper analyses the cognition status and laws of freshmen students, especially the characteristics when constructing programming knowledge, and explores the reform of programming teaching from many aspects.

The course of C programming language is offered in the first semester of freshman year, and students at this stage have the following characteristics when constructing programming knowledge:

1. **The computer system structure is not learned**: The textbook *Computer Systems – A Programmer's Perspective* introduced the intrinsic conceptions of computer system. Furthermore, it explained how these conceptions impact correctness, practicality and performance of the programme (Bryant, David Richard & David Richard, 2003). In fact, from the perspective of the computer system, the representation of information, machine-level representation of the programme, processor architecture and memory hierarchy have a profound impact on programming. It is impossible for a freshman to understand the complete computer system knowledge hierarchy; however, we cannot avoid the *blockers* in the cognitive process of students. In order to let the students understand the basic process and principles of programme execution, it is necessary to describe the key issues for the students in a vivid manner.
2. **Programming ideas are not established**: For most freshmen, programming is a brand new problem. It is even hard to compare with previously learned knowledge. Consequently, **i**t is not conducive to knowledge construction through the two-way effect of new and old knowledge. For example, the students know how to write a mathematical equation, but do not know how to convert it into the source code; for non-math problems, the gap is even larger. In fact, the previous knowledge like number theory, Boolean algebra, permutations and combinations are all important mathematical logic foundations for programming. It will concur with students cognitive laws if we mine the *old knowledge*, design the content of courses in a step-in-step manner and use the two-way effect of new and old knowledge to construct knowledge.
3. **Engineering literacy is not trained**: Learning programming language is a new challenge for freshmen. It is the best time to train good habits and ways of thinking. For specialised computer practitioners, it is necessary to be sensitive to new knowledge and new tools, and have a good sense of rules. We should deliberately train students in these aspects at the beginning, rather than wait until their bad habits have formed and take extra classes after they leave the school. In general, this *preconceptions* approach in terms of habits will be more conducive to the cultivation of professionalism.

2. Method

In the reform of programming teaching, Ismail, Ngah and Umar (2010) established a *needs assessment* mechanism. Guo, Li, Lu and Guo (2019) adopted a method that combines students' own majors. However, these methods are only suitable for non-computer majors, and cannot meet the requirements of helping them establish a *system-level vision.* Some scholars pay attention to the research of computational thinking (Ding, Wang, Zhao, Yan & Yang, 2020; Israel, Pearson, Tapia, Wherfel & Reese, 2015), but high school education experts have found that computational thinking in programming can help students build interesting ways of thinking (Garcia-Penalvo & Mendes, 2018). This inspires us to think backwards and dig out the cornerstones of the knowledge system in middle school that are helpful to our programming knowledge construction. In this work, based on the cognitive status and laws of freshmen students, we follow the process of *breaking barriers -> step by step -> curing abilities* to help students build programming thinking, build knowledge systems and train engineering literacy.

2.1. Establishment of programming thinking based on computer system perspective

Von Neumann's structure that combines programme instructions and data for storage is the basis of modern computer system structure. The execution of the programme is actually a cycle for fetching instructions, parsing instructions and executing instructions. From the perspective of the computer system, introducing the basic execution process and principle with the simple *hello world* programme is helpful for students to clear the obstacles of programming thinking. At present, there remain the following issues to be solved:

1. **How to make the computer understand our intention?** In other words, how does the computer understand our source code in programming grammar? The programming course usually provides an integral developing environment (IDE), editing, compiling, linking and running methods. In fact, this procedure is converting the source code into machine language, but it is hard to understand by students. Consequently, observing the input and output of *Compilation preprocessing–compiling–assembling–linking* by a simple programme is helpful for the students understand the basic of computer *conversation*.
2. **How the computer works?** Based on the system structure of programme instruction storage, programme execution usually follows the loop process of *fetch–decode–execute*. With a simple programme like *hello world*, through animation demonstration and observation of changes in registers, students can understand the entire execution process.
3. **How the computer stores?** Computers follow the binary storage management mechanism, which includes managements of primary storage and secondary storage, static storage and dynamic storage, heap area and stack area. Guiding the students to observe the storage location and life cycle is helpful in understanding how the computer obtains instructions and data and processes them. For example, stack area is managed by the system. The storage in the stack area will be automatically recovered when the function returns and local variables of the function will be invalid. It cannot be accessed even through indirect access by the calling function; on the other hand, heap area is managed by the programme, i.e., if the custom function calls the storage unit requested by malloc function and the corresponding storage unit is not released at the end of the function operation, it can be accessed indirectly by the calling function.
4. **How can a computer work fast and well?** Although *performance* is hard to be attracted by a freshman of programing design, it is indeed a necessary ability for an excellent software practitioner.

Solving the aforementioned issues will reveal the mystery of computers and eliminate fear for students. This enables students to understand the programme design more clearly.

2.2. Teaching structure design based on knowledge construction

The knowledge construction mechanism is the interaction mechanism between the knowledge construction subject and the cognitive object in the cognitive process (Li & Wang, 2019; Ma, Yue & Wang, 2021; Zhang, He, Shang, Xia & Hu, 2020). Knowledge learning can be divided into three stages: knowledge generation and understanding, knowledge integration and deepening and knowledge application and transfer. All stages of knowledge learning follow the two-way construction process of new and old knowledge. In view of the characteristics of programming courses, we will conduct the following strategies:

1. **Fundamentals of mathematical logic mining**: The knowledge that students learned in middle school can actually be used as the basis of mathematical logic established by programming ideas. For example, Boolean algebra can help students understand logical operations; compound functions can help students understand nested function calls; and permutations and combinations can help students understand enumeration algorithms. By mining the old knowledge and exploiting new knowledge, the two-way construction of new and old knowledge can be implemented to help students quickly establish the basic framework of programme design ideas.
2. **Definition of step-by-step study cases**: The next issue is how to fulfil the teaching structure. The main stream of *C programming language* consists of the basic, advanced and high-level component. The basic component contains constants, variables, sequence selection loop etc.; the advanced component contains arrays and functions; and the high-level component contains indirect access and structure. For each component, it is suggested that the design step-and-step study cases analyses each question deeper; this will help the students establish their own effective knowledge structure.
3. **Open-minded inspiration**: After having a certain understanding of a certain knowledge sector, students need to be inspired to think independently and innovatively. It can be achieved by *one problem with multiple solutions.* For example, for the classic *eight queens* problem, we can inspire the students in both data structure and algorithm; in view of data structure, the chessboard is usually a 2-D array, but a 1-D array is also a feasible solution; in view of algorithm, the problem can be solved by a nested loop, and the computational complexity can be reduced by appending constraints. Indeed, recursion is the default solution to the *eight queens'* problem. In sum, for these contents, it is suggested to inspire students and to encourage study themselves, because *there is no fixed method, only an open mind.*
4. **Subject field of vision extension**: We suggest that to expand students' subject field of vision; the course content itself should also be brave to get close to the latest technology. The next two examples are related to graphics rendering and machine learning, respectively. 1) Most students like to play digital games; they may be familiar with ray tracing in games, as this is one of the classic rendering algorithms in graphics and it can be implemented by C language; 2) perceptron is the basic union of artificial neural networks; it is a single neuron model and can also be implemented by C language. In fact, through the above study cases, the students cannot thoroughly understand the theory behind, the later *experiential experiments* show robustness of C language.

Based on the above four strategies, we obtain a spiral knowledge construction model, as shown in Figure 1.

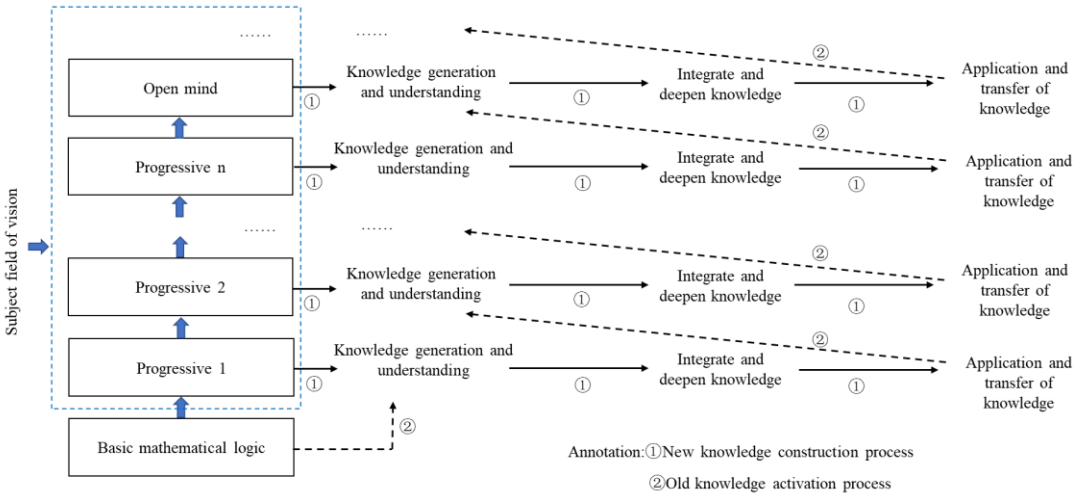

Figure 1. Spiral knowledge construction model

We construct the knowledge system with the main stream and each knowledge sector with the mathematical logic. Each new knowledge is based on the previous knowledge in a step-in-step manner. The detailed stages are as follows:

1. **Knowledge introducing and understanding**. The new knowledge is constructed by long- and short-term memories of the activated new information and its related knowledge.
2. **Knowledge integration and deepening**. Review the new knowledge to prevent forgetting the long-term memories. Furthermore, variation exercises and review for integration and deepening are necessary.
3. **Knowledge application and transfer**. Through the repeated extraction and application of knowledge to achieve analogy to similar problems

2.3. Engineering literacy training based on the background of new engineering

Professional literacy training is important for the freshmen of the programming language (Li, Hei, Wang, Li & Qu, 2021). During the teaching processing, we should deliberately and *actively* help students build a good awareness of the following norms: cognitive tests, using tools, discovering new knowledge, etc. The details are as follows:

1. **Habituation to standard**: It is easy to assess a programmer by only a few codes. The code of a good programmer is readable and maintainable. The standard of a system contains not only the code but also the document. *Developing a habit in 21 day* helps students to develop standardised habits through the first 3 weeks and will have a positive impact on their entire career (Ma et al., 2021).
2. **Awareness to test**: Software testing is a professional course offered in the senior year, but without basic test idea, it will not be able to find potential errors in the programme. Although freshmen students cannot systematically learn the complete theoretical knowledge and practical skills of the test, we can help them gain a basic understanding of the test through study cases. For example, testing of quadratic equation in one variable programme can be designed based on basic mathematical knowledge.
3. **Proficiency to programming tools**: A freshman begins to use IDE at first in programming; however, editing, compiling, linking, running, debugging, finding help and setting compiler parameters need a learning process. Most freshmen ignore management tools; they usually use explorer to name files or folders by date to distinguish different versions. Even worse,

some of them do not manage versions; they delete their written programmes. Consequently, we should train students to use version management tools correctly.

We suggest teaching according to students' cognitive characteristics and cognitive laws, adopting appropriate strategies to integrate these contents into teaching organically. Let students do while learning, learn by doing and organise practical teaching content according to the process of *experience–verification–exploration*. *Experience* means the students observe different outputs of the programme through different parameters; *verification* means verification of unit knowledge points, usually a simple programme; *exploration* means comprehensive applications that we can follow in the advanced organisation of *Fill in sentence -> Fill in block -> Write programme -> Write project*. As shown in Figure 2, for *experiential and verification experiments*, by using an online judge not only makes students view the completion of the code once they complete the code, but also effectively saves the teacher's inspection time; for *exploration experiments* we use project management platform for process management where teachers can publish advanced tasks and development specifications in the form of documents and give sample projects and test the submissions of different versions of students. Students can complete tasks on the platform as required, and each completed task can be merged into the main branch to form a new version. No matter what kind of practical teaching, the given documents, procedures and projects must strictly comply with the software engineering specifications. This helps creating a good language environment for students to learn programming.

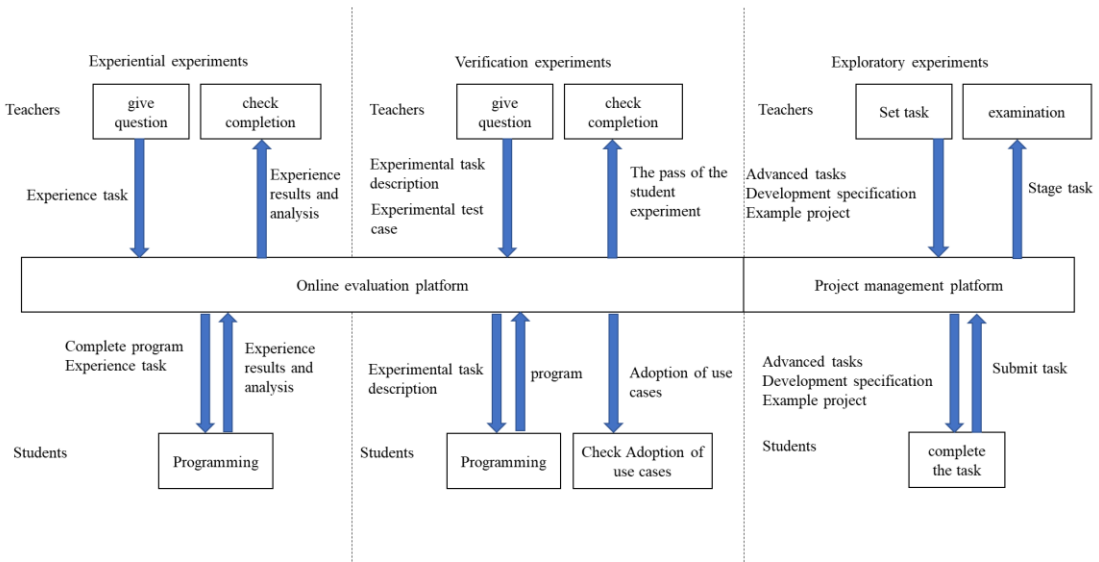

Figure 2. Practical teaching mode based on *experience–verification–exploration*

3. Results

We have been conducting teaching reform for the last 2 years of C programming language teaching. The credit hours of the experiment course have been increased from 24 to 32, and all the class scales are reduced and each class has a special teacher responsible for experimental guidance. This reform is in favour of *exploration experiments*, i.e., teachers establish projects from the programme management platform and invite students to complete the tasks. The following is an example of the teaching data of C programming language in the fall of 2019.

Observing from the completion of the experiment:

1. 100% of the students complete the *experiential experiments*;

2. 100% of the students complete the basic experiments of *verification experiments*, 91% of the students complete advanced experiments and 80% of the students complete high-level experiments;
3. 92% of the students complete basic experiments of *exploration experiments*, 80% of the students achieve 85% comprehensive completion, 60% of students have carried out personalised development and 25% of them are excellent in development.

Table 1 shows the comparative data of the two semesters. From the point of view of the final exam, students have a certain degree of improvement in the integrity of the knowledge system and the effective improvement of practical ability, and the score of the test paper of the same difficulty is increased by 6%. From the perspective of teaching evaluation, the scores of the students in the two classrooms are 95 points (out of 95 points), which is higher than 0.8 points in the C programming language classroom in the fall of 2018.

Table 1. Comparison of the two semesters

Item		Score	
		2018	**2019**
The component of the final exam	Basic (40%)	86.4	89.3
	Advances (40%)	82.5	87.0
	High level (20%)	60.2	69.4
	Comprehensive	79.6	84.4
The teaching evaluation		94.2	95.0

In sum, the programme design teaching reform based on students' cognitive law achieved the following results:

1. Respect the laws of cognition and construct a knowledge system. The knowledge system is constructed by a computer system. This reform breaks cognitive barriers, digs out old knowledge, expands new knowledge, builds the integrity of the knowledge system, ensure the depth and breadth of knowledge, allows most students to learn knowledge from the curriculum and contributes to the achievement of inclusive education.
2. Improve practical ability and cultivate innovative thinking. Practicing the concept of learning by doing and letting practical teaching help theoretical teaching; the spiralling knowledge structure progresses layer by layer, with multiple solutions to one question, effectively guiding students to think and solve problems and cultivate students' innovative thinking.
3. Strict process management and strengthen professionalism. Strict management of the teaching process through the online evaluation platform and project management platform, which effectively guarantees the learning time and efficiency, repeated testing and reduces plagiarism; the management process pays attention to software engineering standards and improves professionalism of students.

4. Discussion and conclusion

The programme design teaching reform based on students' cognitive law has been designed, practiced and improved, and the experimental results show that the work achieved satisfactory results. Based on the laws of students' cognition, through the establishment of programming thinking based on the perspective of computer systems, the design of the teaching structure based on knowledge construction and the engineering literacy training based on new engineering backgrounds, a spirally ascending complete curriculum theory knowledge system has been constructed. *Experience–Verification–Exploration of the* practical teaching mode, focusing on cultivating students' sense of innovation and professionalism, will help cultivate software talents that meet the needs of the

industry. Among them, knowledge construction thinking based on mathematical logic mining, step-by-step study cases, open-minded inspiration and subject field of vision extension guides teachers to help students build a complete knowledge system with high quality and innovation; engineering training based on habituation to standard awareness to test and proficiency to programming tools has effectively cultivated students' engineering literacy. The practice shows that compared to previous teaching methods, this systematic cultivation method is very effective from the first programming language learning in the first year of university.

In the future, we will conduct extensive research on knowledge construction, especially on the different ways of organising teaching according to different knowledge and knowledge construction subjects, in order to deepen students' theoretical knowledge and improve programming ability.

Acknowledgement

The authors gratefully acknowledge the research funding from the following programmes:

- Teaching Reform Project of Huazhong University in 2018 (2018021).
- National Natural Science Foundation of China Instrument Project in 2019 (61927801).

References

Bryant, R. E., David Richard, O. H. & David Richard, O. H. (2003). *Computer systems: a programmer's perspective* (vol. 2). Upper Saddle River, NJ: Prentice Hall.

Chen, L. J. & Zhou, H. B. (2017). *Teaching reform based on curricular integration of C programming language & data structure for cultivation of practical programming skills* (pp. 1194–1205). In Computer Science and Technology: Proceedings of the International Conference (CST2016). doi:10.1142/9789813146426_0136

Curricula, C. (2001). The joint task force on computing curricula. *IEEE Computer Society Association for Computing Machinery*.

Ding, S., Wang, P., Zhao, K., Yan, Z. & Yang, X. (2020). Research on project-based teaching oriented to computational thinking ability development. *Modern Educational Technology,* (9), 49–55. doi: 0.3969/j.issn.1009-8097.2020.09.007

Garcia-Penalvo, F. J. & Mendes, A. J. (2018). Exploring the computational thinking effects in pre-university education. *Computers in Human Behavior.* doi:10.1016/j.chb.2017.12.005

Guo, T., Li, Y., Lu, M. & Guo, Y. (2019, August). *Exploration of new teaching method applied in C++ course combined with students' majors* (pp. 903–906). In 2019 14th International Conference on Computer Science & Education (ICCSE). IEEE. doi:10.1109/ICCSE.2019.8845412

Ismail, M. N., Ngah, N. A. & Umar, I. N. (2010). Instructional strategy in the teaching of computer programming: a need assessment analyses. *TOJET: The Turkish Online Journal of Educational Technology, 9*(2). doi:10.1177/0092055X10364615

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M. & Reese, G. (2015). Supporting all learners in school-wide computational thinking: a cross-case qualitative analysis. *Computers & Education, 82*, 263–279. doi:10.1016/j.compedu.2014.11.022

Li, H. & Wang, W. (2019). In-depth online collaborative knowledge construction through the lens of self-organizing. *Distance Education In China,* (1), 47–57. doi:10.13541/j.cnki.chinade.20190114.001

Li, W., Hei, X., Wang, L., Li, B. & Qu, X. (2021). Experimental teaching of C language programming under the background of new engineering disciplines. *Computer Education,* (07), 188–192.

Ma, Z., Yue, Y. & Wang, W. (2021). Analysis of collaborative learning engagement based on multimodal interaction information. *Modern Educational Technology, 237*(01), 47–53. doi:10.3969/j.issn.1009-8097.2021.01.007

Qin, L. & Tan, Z. (2016). Strengthen the system ability and promote the cultivation of independent and controllable talents in information industry. *China University Teaching,* (7), 37–43. doi:10.3969/j.issn.1005-0450.2016.07.008

Xinogalos, S., Pitner, T., Ivanovic, M. & Savic, M. (2018). Students' perspective on the first programming language: C-like or Pascal-like languages? *Education and Information Technologies, 23*(1), 287–302. doi:10.1007/s10639-017-9601-6

Zhang, S., He, J., Shang C., Xia, D. & Hu, Q. (2020). Cognitive input analysis model and application for online learning collaborative knowledge construction. *Journal of Distance Education, 38*(4), 95–104.

Zhao, Y. & Chen G. (2020). Research on C++ programming teaching based on cognitive task analysis. *Computer Education,* (11), 147–150. DOI: 10.16512/j.cnki.jsjjy.2020.11.036