Research On the Teaching Method of Programming Course by Using Computational Thinking

Lei Chen[1], Xuebo Zhang[1] *

[1]*School of Space Information, Space Enigneering University, Beijing, China*
my_shitou320@163.com, 178140615@qq.com

Abstract:
The programming course is an important public course for undergraduate, which mainly cultivates students' ability to solve and analysis professional problems by using computational thinking. Taking C language programming course as example, this paper proposes the teaching method of programming course which is oriented by computational thinking, and carries out research from four aspects: classroom teaching, experimental teaching, course assessment and stratified teaching. The teaching practice shows that compared with the traditional teaching mode, the teaching method incorporating computational thinking can effectively improve students' learning effect.

Keywords: *program design course; computational thinking; stratified teaching.*

1 INTRODUCTION

Computational thinking is one of the three major scientific thinking for human beings to understand and transform the world. Its concept was first systematically put forward by American computer scientist Professor Zhou Yizhen in 2006. She believes that computational thinking is a series of science thinking methods that use the basic concepts of computer science to solve problems, design systems, and understand human behaviour. Its essence is abstract and automatic [5]. Computational thinking can help people extract the necessary details from real problems, describe problems in a way that computers can understand, implement the problem solving process automatically. Computational thinking includes logical thinking, decomposition, generalization, pattern recognition, modeling, abstraction, automation and evaluation, among which abstraction and automation are the core concepts of computational thinking. Computational thinking has been highly valued by China's Computer Education Research Association. In the Joint Statement of The Development Strategy of Computer Basic Teaching issued by the Nine Universities Alliance (C9) in China, the cultivation of college students' computational thinking ability is taken as the core goal of college computer basic teaching [2], which has since opened the prelude of the teaching reform of computer basic courses oriented by computational thinking. It also lays the core position of the cultivation of computational thinking ability in the teaching of basic computer courses in universities [1] [4].

Programming design is a basic computer course for science and engineering majors, which cultivates students' ability to solve and analyse professional problems by writing programs from the level of program language. The study of computational thinking in programming courses has aroused plenty of scholars' interest. For example, Chen (2011) systematically summarizes the training points of computational thinking designed in each chapter of C language programming course. Hong (2014) explains the construction and practice of C language from the aspects of course construction objectives, teaching content and methods, resource construction, co-construction and sharing, etc. Ye (2017) [7], Yu (2011) [8], Zhang (2012) [9], Wu (2011), Su (2012) [6] explores the teaching reform of programming courses from the perspectives of case design and project process, but ignores the discussion on the nature of computational thinking. How to integrate computational thinking into programming teaching, so that students can use computational thinking such as abstraction, automation and so on to solve the problems in the field of analysis is a challenge facing the teaching reform under the new situation. This paper takes C language programming course as an example, and on the basis of summarizing the corresponding relationship between computational thinking and course knowledge points, studies how to integrate computational thinking into classroom teaching, experimental teaching and

B. Fox et al. (Eds.): IC-ICAIE 2022, AHCS 9, pp. 383-389, 2023.
https://doi.org/10.2991/978-94-6463-040-4_58

course assessment. At last, this paper introduces the hierarchical teaching method which is integrated with computational thinking.

2 THE CLASSROOM TEACHING METHOD INTEGRATING COMPUTATIONAL THINKING

As the main field of teaching, classroom teaching determines the teaching effect to a great extent. By integrating computational thinking into the classroom teaching of programming course, students can contact and feel computational thinking in theoretical learning, thus laying a solid foundation for correct use of computational thinking in programming practice. According to the textbook "Programming" edited by Professor Tan Haoqiang, we have integrated the abstract, automatic, exhaustive, divergent, reverse and generaliztion thinking of computational thinking into each chapter of knowledge points in classroom teaching, forming our own teaching characteristics.

2.1 Abstract Thinking

For C programs, abstraction is embodied in using symbolic systems to accurately and strictly describe the solution of the problem, which runs through all the chapters of C programming courses. We integrate computational thinking into the process of explaining theoretical knowledge from the following aspects.

(1) The program is the abstraction of the actual system, it abstracts the characteristics of the system to the operation object, and abstracts the functions of the system to the function. C program development process is the process of system modeling. (2) C abstracts the value range of features and allowed operations into data types. Data types and operators are the vehicles for system abstraction. Each data type has various value ranges and operations. For example, course grades are floating point numbers ranging from 0 to 100, which can be used for arithmetic and logical comparison operations. The course name is a string of data, so you can't do arithmetic on it. (3) Constant is the abstraction of invariable features in the system, such as PI; A variable is an abstraction of the characteristics in the system that can be changed, such as the radius of a circle. (4) the selection and loop structure in the program are abstractions of the solving steps. (5) Functions in C language are abstractions of functional modules. Mutual calls between functions shows the complexity of the system. For example, a function that evaluates a definite integral must call a function that implements fixed-point evaluation of a variable expression. If the variable expression is complex and contains trigonometric functions, multi-level calls will be formed. When a function needs to call itself during execution, it forms a recursive call. (6) One-dimensional and multidimensional arrays are abstractions of one and more system properties with the same data type, respectively. For example, a one-dimensional array is used to describe the scores of all students in advanced mathematics, and a two-dimensional array is used to describe the scores of all students in advanced mathematics and English. (7) Structure is the abstraction of several system attributes with different data types, such as using structure to describe the name, height, age and other characteristics of a student. (8) The file is the abstraction of system properties, so as to facilitate storage and automatic read and write operations.

By integrating abstract thinking into the above knowledge points, students can fully understand abstract concepts and usage scenarios.

2.2 Automation Thinking

Another essential aspect of computational thinking is automation. Automation refers to the imposition of some operation on the various elements of the symbolic system modeling and the automatic execution of some sequential or non-sequential structure. In the theoretical knowledge points of C program design course, the algorithm flow reflects the automation thinking. Data type, operator, sequence structure, selection structure, circular structure, function, array, pointer, structure and file are all automatic execution units of C program.

In order to enhance the students' understanding of automatic thinking, we emphasize that the automation of the c language program is reflected in the following two aspects: one is that the internal statements of the function are automatically executed under the support of the EIP register addition operation, and the second is that the mutual call of the function is realized through the automatic jump in the inside stack. The program statement is the execution object of the computer automation, and the combination of the statements constitutes the function. The essence of the program's execution is to automatically execute the statements in a certain function, and automatically jump between multiple functions. Automatic execution requires the support of computer software and hardware, and the operating system provides the software base of the program's automatic execution, and the computer hardware structure and the thought of program storage provides the hardware foundation for the automatic execution of the program. The operating system coordinates memory, CPU, and external hardware resources to execute the process.

The automation process actually reflects the algorithm flow of C language program to solve practical problems. An algorithm is an execution step to solve a problem. Generally speaking, before programming with C language, it is necessary to design algorithms, that is, to conceive ideas and steps to solve practical problems.

When these steps are syntactically translated into C statements and formed into a complete program, the operating system can automatically execute them to complete the corresponding work. In fact, automation is one of the core reasons why computers can solve problems so efficiently.

Through the introduction of the above knowledge, we connect the automatic thinking with computer hardware system, von Neumann program storage principle, operating system resource scheduling and other knowledge in the classroom teaching, so that students intuitively and profoundly master the professional basic knowledge of the computer discipline system.

2.3 Decomposition Thinking

Decomposition thinking method is a thinking method that considers problems from the systematic perspective of things and aims at obtaining the best solution (Dong, 2019). The thought of decomposition programming is the perfect embodiment of system thinking. We guide students to focus on the relationships between software system and local modules, internal system and external systems, and the mutual restriction between modules in system design. In this way, the complex problem is decomposed into several sub-problem modules that are easier to solve at a macro level. Microscopically, appropriate methods are selected to solve each sub-problem based on its characteristics.

In the project practice of C program design course, we require students to carry out simple information system design, so as to further strengthen their understanding and mastery of decomposition thinking. For example, in the demand analysis stage, we require students to consider the software function division, the relationship between each function module, the contradiction between hardware and software, the contradiction between software development and user demand and other factors from the perspective of the system. The teaching practice proves that the cultivation of decomposition thinking can effectively improve students' ability to solve problems from a macroscopic perspective.

2.4 Logistic Thinking

Logistic contains many factors, we take the exhaustive thinking as example to demonstrate the teaching way we use to enhance our students' logistic thinking ability. The Exhaustive method is the use of computer operation speed, high precision characteristics, to solve the problem of all possible situations, a leak to find out the answer to the requirements. Exhaustive thinking is a good way to solve problems that have no rules, such as digital cryptography, which yields results regardless of time. Exhaustive thinking in C language loop, array has a wide range of applications, such as daffodils, prime number, maximum, cryptography and other problems can use the exhaustive method.

We guide students to master the positive and negative aspects of exhaustive thinking through examples, and guide them to understand the advantages and disadvantages of exhaustive thinking through the complexity of algorithm, so as to make rational use of exhaustive thinking in future study and work.

2.5 Generalization thinking

Generalization thinking is an very important composition of computational thinking, which can be trained by the way of "one problem and many solutions". In class, for the same problem, different groups of students will come up with different solutions. Through the comparison of different solutions, students' generalization thinking is fully displayed. For example, to find the maximum value of 3 numbers, four methods were concluded after groups of discussions:

Method A: Using single branch structure.

```
max=x;
if(y>max) max=y;
if(z>max) max=z;
```

Method B: Using double branches structure

```
if(x>y) t=x; else t=y;
if(z>t) max=z; else max=t;
```

Method C: Using multi-branches structure

```
if(x>y&&x>z) max=x;
else if(y>z)max=y;
else max=z;
```

Method D: Using nested branch structure

```
if(x>y)
   if(x>z)max=x;
   else max=z;
else
   if(y>z)max=y;
   else max=z;
```

These four methods reflect different ideas. Although there are still some imperfections, they are all the results of independent thinking and full discussion by students. Then, the teacher asked which method was more suitable for a large amount of data, in order to guide students to generalization thinking and deepen their understanding of the problem.

3 EXPERIMENTAL TEACHING METHOD INTEGRATING COMPUTATIONAL THINKING

C language is a very practical course. In this section, we will take abstraction, automation and decomposition as examples to introduce how to integrate computational thinking, such as into experimental teaching.

3.1 Abstraction and Automation

Abstraction and automation are the essence of computational thinking. Different from classroom teaching, in experimental teaching, we focus on improving students' understanding of abstraction and automation thinking from the perspective of using programs to solve practical problems.

For example, C language data types are methods for abstracting, representing and processing information in the real world. We guide students through the following 3 steps to use abstract and automated thinking to design systems. First, according to the requirements of the system, the transformation of the system description from the real world to the information world is realized. Through analysis, abstract thinking is used to obtain a conceptual model of the data that the system needs to process. Second, realize the mapping from the information world to the machine world. Convert the conceptual model of the data to be processed into a data structure supported by the C language grammar rules. Finally, use C language to write a program, and the computer automatically realizes data processing. The specific process is shown in Figure 1

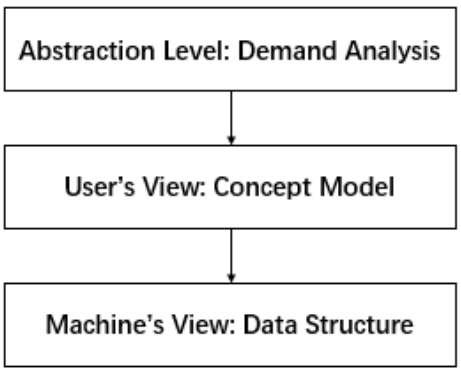

Figure 1: Abstraction and Automation in Programming

In the teaching process, teachers introduce abstraction and automation methods through examples, guide students to gradually understand abstraction and automation thinking methods, and improve computational thinking ability in the process of solving problems.

3.2 Decomposition

The decomposition method of computational thinking is to decompose a complex problem reasonably, study the solutions of each sub-problem separately, and finally solve the complex problem as a whole after summarizing [3].

In the process of experimental teaching, we instruct students to divide the program design into 7 steps. (1) Analyze the problem to determine the required data structure, (2) Assign initial values to the variables involved in the operation, (3) Use the three basic structures of sequence, selection, and loop to complete the logical expression of problem solving, (4) Determine the output, (5) Draw a flowchart, (6) Write the program, (7) Debug on the computer. The tasks of each stage are independent of each other, with clear completion signs. The result of the task of the previous stage is the premise and foundation of the task of the next stage, and the task of the latter stage is the deepening of the task of the previous stage.

For example, when writing a program to determine whether a number is prime, follow the steps below.

Step 1: Analyze the problem and determine two integer variables *m* and *i*, where m is the number to be judged as the dividend, and *i* is the divisor.

Step 2: Assign initial values to the variables involved in the operation. Input the value of m from the keyboard, scanf ("% d", &m); A prime number is not divisible by any other number except 1 and itself, so the initial value of *i* is determined to be 2.

Step 3: Use the three basic structures to solve the problem, which is the key to solving the problem. The judgment of a prime number is to divide m by every number in 2 ~ m-1. If each number cannot be divisible, it means that m is a prime number. Otherwise, as long as there is a number that is divisible, then m is not a prime number. Use loop structures to solve problems.

Step 4: Output the result. According to the analysis of step (3), if every number in 2 ~ m-1 is not divisible by m, after the cycle ends i>=m, the output m is a prime number. If m is not a prime number, i<m, the output m is not a prime number, obviously the output of the result needs to use the selection structure.

Step 5: Draw the flow chart as shown in Figure 2.

Step 6: Write the program

```
#include<stdio.h>
void main(){
    int m, i;
   scanf("%d", &m);
   for(i=2; i<m; i+{
```

```
        if(m % I == 0) break;
        if(i>=m)
           printf("m is a prime.");
        else
           printf("m is not a prime.");
    }
}
```

Step 7: Debugging on computer.

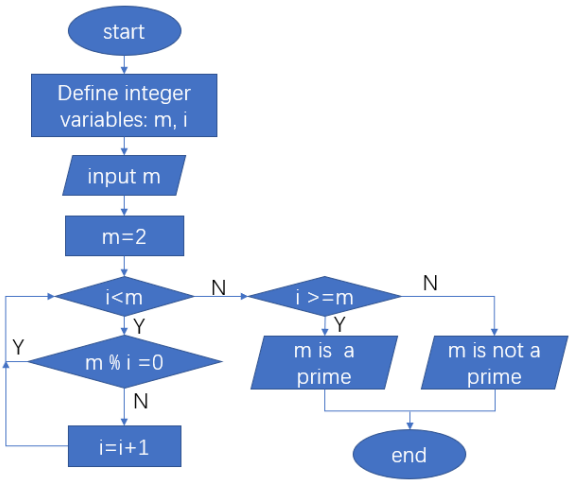

Figure 2: The Flowchart of Prime Number Judgement

Through example explanation, students not only master the basic knowledge of programming language, but also practice the methods and steps of programming, so as to avoid students being bored with learning because they feel that the grammar knowledge is scattered and complicated. Our teaching practice shows that using decomposition method in experimental classes can improve students' ability to comprehensively apply what they have learned and design programs to solve practical problems.

The optimization features of computational thinking can also be well integrated into the C programming process. There may be multiple algorithms for any problem. When designing an algorithm, we must not only solve the problem, but also consider the time complexity and space complexity of the algorithm to find the optimal algorithm. For example, in the above prime number judgment method, the loop structure is used to solve the problem part, and the number of loops is m-2 times. In fact, it can be simplified. m does not need to be divided by every integer between [2, m-1], but only needs to be divided by integers between [2, $\sqrt{m}$]. Therefore, the program can be modified as follows to reduce the number of loops.

4 COURSE ASSESSMENT METHODS INCORPORATING COMPUTATIONAL THINKING

In view of the computational thinking training requirements of the C language programming course, we have carried out the following two reforms on the final assessment method of the course.

First of all, the assessment method must be changed from mainly examining the grammar knowledge of the language to mainly examining the students' system modeling ability and algorithm design ability. The examination of system modeling ability is mainly aimed at the abstract characteristics of computational thinking. For example, various system characteristics can be given to test students' ability to describe using data types. The test of algorithm design ability is mainly aimed at the automatic characteristics of computational thinking. Of course, C programming courses involve simple algorithms such as iteration, enumeration, and sorting. These algorithms can be encapsulated into various functions, and the automation process can be understood by studying the execution order of the statements in them, and the parameter passing during function calls. Therefore, we recommend focusing on simple algorithm design and function calling, and highlighting the assessment of function interface design and algorithm process design.

Secondly, increase the score of programming questions in the final exam paper, the proportion of which will increase from 30% to 50%, and the setting of programming questions should be done gradually to assess students' ability to use C to abstract practical systems, system modeling ability and Algorithm design ability is the main purpose.

5 EVALUATION

In recent years, the teachers of the university computer basic teaching team have implemented the computational thinking-oriented programming teaching method and achieved gratifying results, which are mainly reflected in the following three aspects.

(1) The final exam scores of the programming course have improved greatly, and the make-up exam rate is reduced to 5% when the proportion of programming questions is increased.

(2) Course students have achieved excellent results in national professional competitions such as the "National Green Computing Contest".

(3) The students' interest and enthusiasm for programming in our school have been improved, and the students' programming ability has been generally improved.

6 CONCLUSION

Programming course is an important platform for undergraduates to cultivate computational thinking ability. Through teaching practice, this paper introduces the teaching method of programming course integrating computational thinking in detail. Practices in classroom teaching, experimental teaching, and assessment have proved that our teaching method has achieved good results. In the future teaching research, we will continue to improve the teaching method of programming courses based on computational thinking, and further build an online and offline hybrid teaching model that incorporates computational thinking.

REFERENCES

[1] Chen Guoliang, Dong Rongshe (2011). Computational thinking and university computer foundation education. J. Chinese University Teaching, 2011(1):7-10.

[2] He Qinming, Lu hanquan, Feng boqin (2010). The core task of computer basic teaching is to calculate the cultivation of thinking ability. J. Chinese University Teaching 2010(9):5-9.

[3] He Mingting (2009). The separation of concerns is the methodological significance of computational thinking and software engineering. J. Computer Science. 36(4):60-63.

[4] Hong Bing, Yao Lin, Wu Hangxing, etc (2014). The analyzation of calculation of the design course in c, J. Chinese University Teaching. 2014(9): 59-62.

[5] Jeannette M Wing (2006). Computational Thinking. J. Communications of the ACM, 49(3): 33-35.

[6] Su Haiying (2011). The practice of programming and teaching of programming in the direction of computational thinking. J. Modern Computer. 2012 (4): 32-34.

[7] Ye Jun, Wang Lei, Han Yuzhen, etc (2017). The construction and practice of the c language in the provincial quality resources sharing course. J. Computer Education, 2017(7):80-84.

[8] Yu Shaobing (2011). The cultivation of computational thinking and programming ability. J. Computer Education. (16): 11-14.

[9] Zhang Yaowen (2012). A study of the teaching method of programming courses based on computational thinking. J. Journal of chongqing electronic engineering college, 21(3):149-150.