Higher Education Research
2022; 7(5): 149-152
http://www.sciencepublishinggroup.com/j/her
doi: 10.11648/j.her.20220705.12
ISSN: 2578-9341 (Print); ISSN: 2578-935X (Online)

Exploration and Implementation of Teaching Reform for Compiler Principle

Yanhua Wang*, Yue Feng, Chunhong Zhang, Ping Luo

College of Electronic Information Engineering, Langfang Normal University, Langfang, China

Email address:
caerus@lfnu.edu.cn (Yanhua Wang), fengyue_92@tju.edu.cn (Yue Feng)

*Corresponding author

To cite this article:
Yanhua Wang, Yue Feng, Chunhong Zhang, Ping Luo. Exploration and Implementation of Teaching Reform for Compiler Principle. *Higher Education Research*. Vol. 7, No. 5, 2022, pp. 149-152. doi: 10.11648/j.her.20220705.12

Received: August 4, 2022; **Accepted**: September 9, 2022; **Published**: September 16, 2022

Abstract: The course of Compiler Principle is rich in theory which have strong logicality. It is an important professional compulsory course for undergraduate students majoring in computer science. Its theory is difficult to understand. The design and implementation of experiments need a certain basis of programing. For the study of theoretical concepts of the course, we propose an inverse derivation method. In addition, aiming at the experimental section, we propose a progressive experimental method to simplify the implementation of comprehensive experiments. Therefore, we construct a mutual feedback teaching system of theory and experiment. With the help of the system, students can not only understand theory easier, but also understand the causes of theory furtherly. Furthermore, the system can lead student to expand their thinking method. Through the trial of the system for students majoring in computer science and technology (Grade 2018) of Langfang Normal University, we found that the students' ability of understanding concepts have been improved to a great extent and their ability of logical thinking have been deepened. With regard to the experimental aspect, their experimental ability have been further improved, and the ability of transformation from theoretical problems to practical applications have been strengthened, which achieved a certain better effect of teaching.

Keywords: Compiler Principle, Teaching Reform, Experiment Reform

1. Introduction

Compiler principle is a compulsory course for majors of computer [1, 2]. It plays an important role in undergraduate teaching [3, 4]. It is a key part of the whole theoretical system. It is indispensable for computer advanced language design system [5]. The course systematically plans the basic principles and the development and implement method of compile program. It mainly includes grammar, lexical analysis, syntax analysis, attribute grammar, semantic analysis, intermediate code generation, symbol table, runtime storage management, code optimization, and object code generation [6]. The course requires students to master the compilation process of advanced language programing, understand and analyze the principles of each stage of compilation in middle layers.

However, the Compiler Principle course is highly theoretical [7]. Its theoretical system is complex, logical and practical, and the theories and practise are closely combined [8-11]. There are common problems on difficult theoretical understanding and difficult practise [12, 13]. It includes a variety of theoretical and abstract concepts and formal descriptions. This requires students to be familiar with formal language and its description method for general problems. Its experiments require design and implementation of algorithms based on understanding basic concepts. During the process of experiments, the construction and verification of basic principles have certain requirements for advanced language programming and implementation.

2. Teaching Reform

2.1. Reform Contents

By concluding the overall framework of the course and analyzing the current situation of teaching and learning, aiming at the typical problems of undergraduate students during their learning process, we try to make some reform.

2.1.1. Theoretical Teaching Methods

The course contains definitions and theorems which are involved in grammar, lexical analysis, grammatical analysis, attribute grammar, semantic analysis and so on. We explore new teaching methods, combine reflective methods, and make extensive analysis on these theories.

2.1.2. Experimental Teaching Contents and Methods

On the basis of understanding theory, the design and development of diversified experiments with different degrees of difficulty are carried out. Design and realize the connection of all experiments. Combine the experimental contents with the compiling process of the advanced language programming flexibly.

2.2. The Proposed Teaching System

We propose an inverse derivation method for teaching and learning of theory. Aiming at the experimental teaching, the progressive experimental teaching method is proposed. By the methods, a complete mutual feedback teaching system of theory and experiment is formed.

2.2.1. Theory Teaching

Definition 1 (The Inverse Derivation Method) With understanding the meaning of theory, start from its results, make derivation in a bottom-up manner. At last, the basic conditions are derived.

We try to understand the concept of SELECT set by Definition 1.

For a Production $A\rightarrow\alpha$ ($A\epsilon V_N$, $\alpha\epsilon V^*$), its SELECT set SELECT ($A\rightarrow\alpha$) indicates the next symbols that it can use. The symbols that the production can produce is determined by its right part α. For $\forall\alpha$, if the symbol set can deduce is M ($M=\{\beta|\ \alpha\overset{*}{\Rightarrow}\beta\gamma,\ \beta\epsilon V_T,\ \gamma\epsilon V^*\}$), then the next symbol that can be used by $A\rightarrow\alpha$ must be β. Thus, $\beta\epsilon$SELECT ($A\rightarrow\alpha$). Because β is the first symbol of the symbol string that can be derived from α, that's First (α)=M. So, SELECT ($A\rightarrow\alpha$)= First (α).

Specially, if $\alpha\overset{*}{\Rightarrow}\varepsilon$ (the derivation length is n) and $\alpha\ \hat{}\overset{*}{\Rightarrow}\varepsilon$ (the derivation length is m, m<n, $\hat{}\overset{*}{\Rightarrow}$means not $\overset{*}{\Rightarrow}$), the next symbol that $A\rightarrow\alpha$ can use includes the result of the previous step, First (α). If First (α)=ε when the derivation length is n, and $S\overset{*}{\Rightarrow}...Ab...$, $b\epsilon V_T$ or $S\overset{*}{\Rightarrow}...A\#$ is satisfied, here $A\rightarrow\alpha$ will not use ε, but use the following symbol of A. That's b or #. Because FOLLOW (A)={b| #}, so SELECT ($A\rightarrow\alpha$)={First (α)-ε}∪ FOLLOW (A).

In addition, for the problem of concept understanding, grammar is described by formal language, which increases difficulties to understanding. And definitions may have strong cohesion. The incomprehensible understanding of the previous theory directly leads to the incomprehension of the following concepts. That is because many definitions have previous definitions, which are described by example in Table 1.

Table 1. *Definitions relation.*

Concepts (name/definition)	Previous Concepts (name)
Language (the language is the collection of all sentences of a grammar)	Sentence
Sentence (if a sentence pattern *x* is only composed of terminators, then *x* is a sentence)	Sentence pattern
Sentence pattern (if a symbol string *x* is derived from the identifier S, then *x* is a sentence pattern)	Derivation. Closure. +Closure
Derivation (*n* direct derivation sequence constitutes a derivation)	Direct derivation. Grammar

For others like concepts in Table 1, we should lay a solid foundation through the consolidation of the previous concepts.

2.2.2. Experiments

We make reform for experiment on 4 aspects to ensure the completion rate and degree.

(i). Diversified Forms

Design various forms, including algorithm verification, module design development, and comprehensive design.

(ii). Import Tools

We will import some sophisticated tools such as Lex and Yacc [14]. Lex is used for lexical analysis, and the self-help construction tool. Yacc is used to make grammatical analysis and semantic analysis.

(iii). Experiments Cohesion

Each experiment has strong independence, which can form its own module. Guide students to connect subsequent experiments. Finally, a simple compiler is constructed.

(iv). Combined with Advanced Language Programming

Through the compiling results of compilers such as GCC, combined with the functions realized in each stage of compiling, thoroughly understand the principle and application of compile system.

We also propose a progressive experiment method, and carry out the total-fraction design and implementation for comprehensive experiments.

Definition 2 (Progressive Experimental Method) Design the overall framework, and carry out the experiment from the local to the overall and from easy to hard.

Table 2. *Predictive analysis table constructing algorithm.*

input: grammar G = (V_N, V_T, P, S), SELECT (P)
output: predictive analysis table M
if G belongs to LL (1) grammar
Define M[m,n], where as m=$\|V_N\|$, n=$\|V_T\|$+1 (including '#')
Initialize, M[0, *i*]=V_T [*i*], M[*i*, 0]=V_N[*i*]
foreach aϵV_T
if aϵSELECT ($A\rightarrow\alpha$)
M[A, *a*]=$A\rightarrow\alpha$
foreach M[*i*, *j*], among which *i*ϵ [0, *m*-1], *j*ϵ [0,*n*-1]
if M[*i*, *j*] is Null
M[*i*, *j*]=error

We take table driven LL (1) analysis for example. Build the overall framework, construct predictive analysis table, design analysis algorithm based on the table, and make syntax analysis. Progressive experimental method is used to ensure

the difficulty and logic are progressive. Firstly, we design an algorithm for constructing a predictive analysis table.

Then, design and understand the syntax analysis algorithm based on the predictive analysis table, which is shown in Table 3. Finally, verify the whole algorithm.

Table 3. *LL (1) analysis algorithm.*

Input: Symbol string χ, predictive analysis table M
Output: Process of grammar analysis
Define stack Z=Null
push (#), push (S),
Initialize, *i*=0, *a*=χ[*i*++]
while *i*<χ. *length*
γ= pop (Z)
if γϵV_T
if γ==*a*
a=χ[i++]
else error
else if γ==‘#’
if γ==*a* break
else error
else if M[γ,a] == {γ→γ1γ2…γk})
push (γ_k), push (γ_{k-1}),…,push (γ_1)
else error

Moreover, experiment is the realization and understanding of theory. Experimental verification can make feedback to theory which can be understood better.

In addition, we guide students to think divergently, and to spread theories to the relevant fields, including the research of other theories, as well as the exploration and combination of theories in other application fields [15]. During the study, keep on exploring ideas and breaking thinking stereotypes, changing the inherent thinking manner and form an open and innovative research-based thinking manner gradually.

3. Effects

In view of the problems encountered in teaching, with analyzing the characteristics of students, the teaching reform method proposed have been used for majors of computer science and technology (Grade 2018) in our school. A preliminary result has been achieved compared with Grade 2017 who did not use this method.

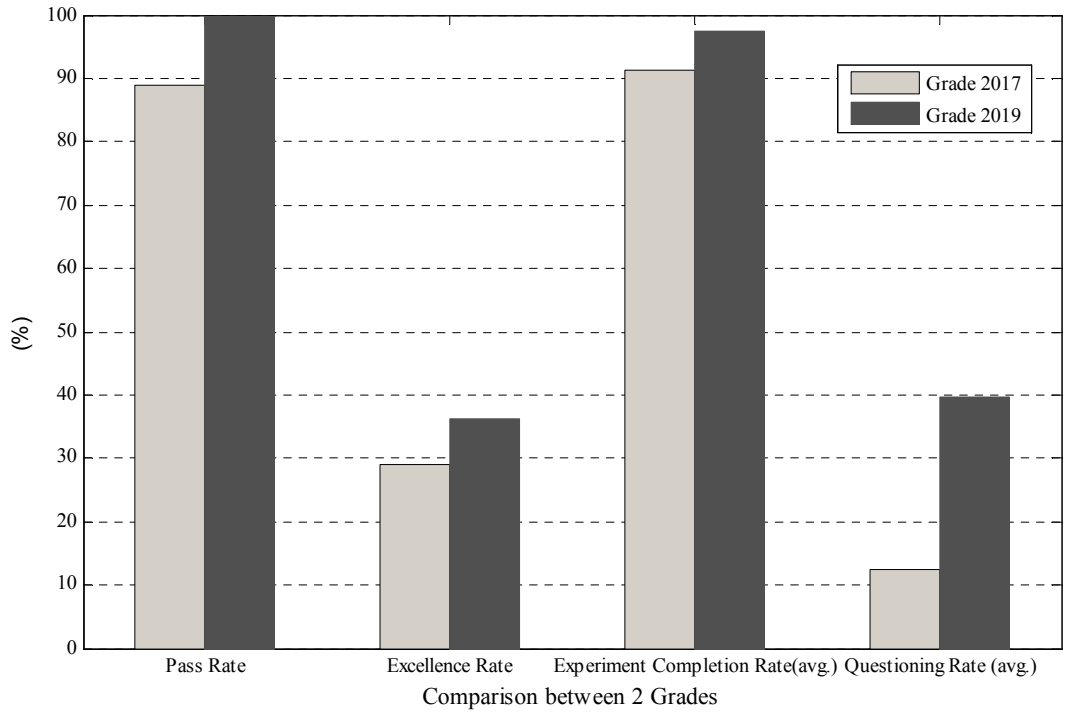

Figure 1. *Teaching effects comparison.*

It can be seen from Figure 1 that the teaching reform method proposed in this paper has achieved a certain effect. In the case of proposition types and difficulty are consistent, the pass rate of Grade 2018 reached 100%, indicating that this method is generally applicable to most students. In terms of score distribution, the excellence rate is significantly higher than that of 2017. The method has a certain effect in raising the grade. In terms of usual teaching feedback, more students participated in questioning and discussion, and the questioning rate increased by 27.26%, which stimulated students' interest in learning. In terms of experiment implementation, the completion of algorithm verification experiments such as lexical analysis experiments has reached 100%, but there are still a few students who cannot complete design experiments independently, who need to further improve the ability of advanced language programming.

4. Conclusion and Future Works

The theoretical system of Compiler Principle course is complex. It includes a variety of theoretical and abstract concepts, formal descriptions and complex experiments. By

studying its overall theoretical composition and analyzing the current teaching situation, aiming at the typical problems in the learning process, we proposed a mutual feedback teaching system of theory and experiment, which provided advanced methods and technologies for principle understanding, analysis, refinement, experimental design and implementation. This system can promote students' logical thinking ability. It plays a significant role in cultivating students' abilities of basic design and algorithm analysis and technical implementation and program optimizing.

Although the teaching system has been proved to be effective, our work still has some shortage which we hope to solve in the future. The teaching system practice is done only for 1 grade and 1 major. To make it widely used in other grades, other majors, or similar courses such as Data Structure and Operating System, we will make more practices for the proposed mutual feedback teaching system. By analyzing the results of examinations, experiments, learning interactions and so on, we would adjust the teaching methods accordingly. Therefore, the teaching system will be stronger and more effective.

Acknowledgements

This work was supported by Langfang Normal University Education and Teaching Reform Project“Research and Practice on the Reform of the Teaching system of Compiler Principle” (No. K2020-38), Langfang Science and Technology Research and Development Plan Project (No. 2021011062) and Langfang Normal University Basic Education Key Project “Application and research of maker education in primary and middle school education and teaching” (No. JCJY202209).

References

[1] G. Qiuyan, “Discussion of Teaching Reform on Compiler Principle Course,” Popular Science & Technology, vol. 158, pp. 174–176.

[2] Y. Jun, “Application of level-division Teaching Thoughts on the Compiling Priciples,” Agriculture network information, vol. 10, pp. 105–108.

[3] H. xi, C. Jia and N. Jian, “Exploration on the Reform of Teaching "Compile Theory,” Journal of Wuzhou University, vol. 31, pp. 85-89.

[4] S. Rui and Z. Xuejun, “Thoughts on the teaching of the course "principles of compilation, "Scientific and technological horizon, vol. 4, pp. 135-135, 162.

[5] Z. Jing, W. Yu and L. Haiyan, “Discussion on the Construction and Reform of Experimental Teaching in Compiler Principle,” Agriculture network information, vol. 3, pp. 133–135.

[6] W. Shengyuan, D. Yuan and Z. Suqin, Compiler Principle, 3rd ed., Beijing: Tsinghua University, 2015, pp. 2.

[7] S. Bing, Y. Haiyan and Z. Li, “Teaching reform and practice of compiling principle course,” Computer education, vol. 2, pp. 73-76.

[8] S. Zhongmei, L. Wenjun and Z. Xiaocong, “Practice and Experience of Teaching Reform for Compiler Principle,” ACTA SCIENTIARUM NATURALIUM UNIVERSITATIS SUNYATSENI, vol. S2, pp. 101-104.

[9] Z. Dongmo and F. Xiwen, “Teaching Practive and Reform of Principles of Compilers Course Design,” Research and exploration in laboratory, vol. 31, pp. 134-137.

[10] H. Li and W. Zhiguo, “Practice of teaching reform in the course of "compilation principle," [J]. China Electric Power Education, vol. 2, pp. 66-67.

[11] W. Tiefeng and Z. Zhichao, “Application of goal decomposition and task driven teaching method in the teaching of Compilation Principle,” Pioneering with Science & Technology Monthly, vol. 012, pp. 85-86.

[12] Z. Ruyan, H. Yanling and Z. Minghua, “Research on teaching methods of computer major courses based on ability training -- Taking classroom teaching and experimental teaching of "compilation principle" as an example,” Industry and Information Technology Education, vol. 11, pp. 52-57.

[13] R. Xiaoqiang, W. Xuemei and T. Xiaohua, “Design and practice of teaching demonstration module of compilation principle based on Python,” Industrial Control Computer, vol. 9, pp. 72-73.

[14] Z. Taisheng and L. Junjie, "Discussion on the teaching reform of compiling principles,” Journal of Chifeng University (Nature Science Edition), vol. 9, pp. 223-224.

[15] Z. Huiping, W. Ting and L. Mengjun, “On the orientation of compiling principle course reflection,”Computer Education, vol. 11, pp. 45-47.