The Progress of Construction of compiling technology course based on different methods

Wang Na[1, a], Zhang ShiMing[1, b]

[1]School of Computer and information, Shanghai Second Polytechnic University, Shanghai, 201209, China

[a]email: wnoffice@126.com, [b]email: smzhang@sspu.edu.cn

Keywords: compiler; construction; automatic; manually;

Abstract. In our former work, we propose a new construction of compiling technology course which divides compiler into two levels: compiler technology and compiler theory. In order to meet the requirement of both two levels, we make a large amount of time to research for the valid experiments. With our practice, the combination of manually and automatic methods is introduced and implemented partially till today. The new combination makes most students be more interested in compiling course since it turns abstraction to specification.

Introduction

With the rapid development of higher education in our country, there has been a complete and special education system. In order to improve employee ratio, lots of university focus on meeting the direct requirements of enterprises. So engineering design and practice has been ignored seriously [2]. To main successful industrialization process in China, keep the sustainable development of national economy, and increase international competitiveness, it largely replies on engineers and technical experts at all levels [1]. Briefly, how to combine the theory with practice is the most urgent need for university education in our country.

Engineering practice teaching is to build up a teaching theory, which values engineering ability training, fundamental theory teaching and professional competence training equally. It also regards engineering teaching as one of the core tasks of personnel training. By means of a comprehensive reform on training mode of Software Engineering, adjusting curriculums setting, enriching the contents of teaching and improving teaching methods, the ability and vocational quality training can be embodied. Moreover, software engineering knowledge is taught through various projects of engineering practice teaching step by step [6].

Compiler technology course is a very important compulsory basic course in professional computer education, and it is also an important branch in the computer system software. The realization of any computer language can be separated from compiler technology. Therefore, as a student major in computer science, it is necessary to learn and master the basic structure and implementation techniques of the compiler for further learning, research and professional work [7].

But, considering current employment situation, the advantage of computer science students in the hardware and software knowledge and application ability is gradually weakened. When competing with students having specific applications and professional background, they face to increasing pressure. Even being employed, few people will engage in the research and development of compiler sys-tem. So the value of compiler's existence as a classic core curriculum of computer science is to be questioned. Especially in some application university, the practice ability of mastering several languages and developing application systems is most paid attention to. Actually, undergraduate education is to solve the students' basic professional ability, capacity for sustainable development which is the essence. The main teaching objective of compiler technology is making students re-understand algorithms and programs on the level of system to enhance the students' system capacity. It is the most difficult course in computer professional courses covering both formal language and abstract automata theory. It is also a comprehensive reflection of knowledge including data structure, programming languages, algorithms and software design. It effectively

trains the abilities of calculated Thinking, algorithm design and analysis, program design and implementation (hardware and software), cognitive, analysis, development, and application of system.

So, a course in compiling techniques is an important part of computing core curricula. The process of compiler construction is supported by well-defined theory and exploit concepts, principles, and software development skills drawn from other related disciplines, such as programming, software engineering, computer architecture and organization, and operating systems. Software engineering techniques can help students understand concepts that underpin the compilation process.

Related Work

In [7], we have introduced our thought of teaching content reform on compiler technology course in application-based university which includes how to cognize the importance of compiler technology and stimulate students' interest. Furthermore, we also developed new methods to meet the above requirement such as web-based teaching method and projects-based experiments.

In [6], a teaching model has been established for compiler principle course through making engineering reform to Compiler Principles course. Ladder-like Iterative teaching method has been introduced. The teaching model and teaching method have provided solutions to the so-called three difficult problems, namely, the difficulties in teaching the compiler principles, the difficulties in understanding and mastering the course, and those in practical development.

[8] shows how object-oriented design patterns represented in UML can be used to both teach type systems and develop the semantic analysis phase of a compiler. The main benefit of this approach is two-fold: better comprehension of theoretical concepts because of the use of notations known by the students (UML diagrams), and improvement of software engineering skills for the development of a complete language processor.

In summary, a few of new methods have been used in daily teaching of compiler technology course. In application-based university, it is necessary for the students to master the construction methods of each step in compiling and know related tools of compiler technology.

Two methods to construct a compiler

In [6] we make certain scale comprehensive design experiments to ensure the effect of practice. We design projects about compiler technology programming carefully. Students will be divided into several groups according to students and their topics of interest. Students selected the leader of each team. Each leader is equivalent to project manager, and is responsible for the entire software project organization and coordination. As mentioned in many books, a compiler system is composed of three main modules that are lexical, syntax and semantics as shown in Fig.1. The experiments are designed from three aspects. When given a simple, a manually construction may be used, otherwise, an automatic construction can be used with some tools.

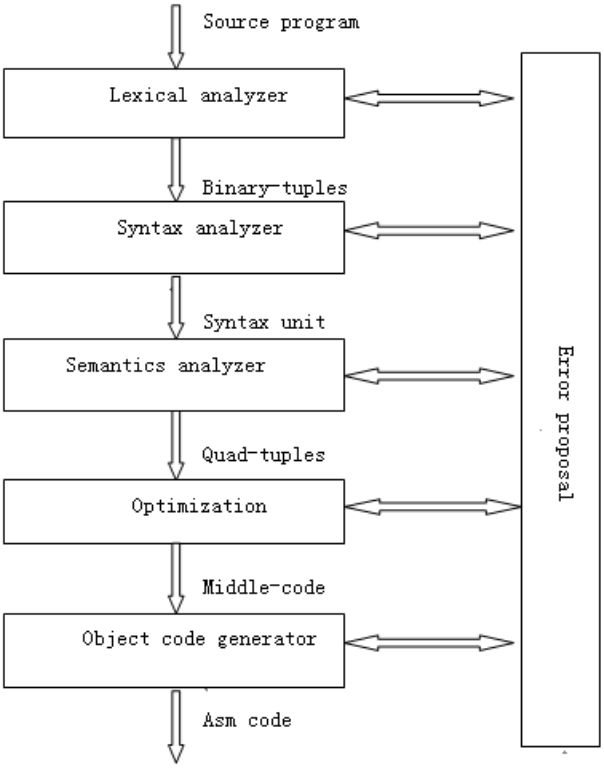

Fig.1. the structure of compiler

Manually construction

In this method, we use algorithms introduced to complement of translating a program based on a certain language to quad-tuple. First, a lexical module is designed to specify the program to a sequence of binary-tuples for the syntax identification. The structure of lexical analyzer proposing function is shown in Fig.2. The semantics analyzer is shown is Fig.3.

Automatic construction

Compiler construction is a widely used software engineering exercise, but the process of compiling is so complex that most students will not be compiler writers. Then tools such as generators and interpreters can help students deal with the complexity.

Flex is a free software alternative to lex. It is a computer program that generates lexical analyzers. It is frequently used with the free Bison parser generator. Unlike Bison, flex is not part of the GNU Project.

Yacc is a computer program for the Unix operating system. It is a LALR parser generator, generating a parser, the part of a compiler that tries to make syntactic sense of the source code, specifically a LALR parser, based on an analytic grammar written in a notation similar to BNF.

With the support of tools, we can complement a compiler according to a model language as Fig.4.

We use two methods respectively to make students master the main theory and algorithms in compiling. Meanwhile, they can also turn to tools when the lexical and syntax is too complex. The introduction of two methods will help students make full use of the knowledge even to their future work.

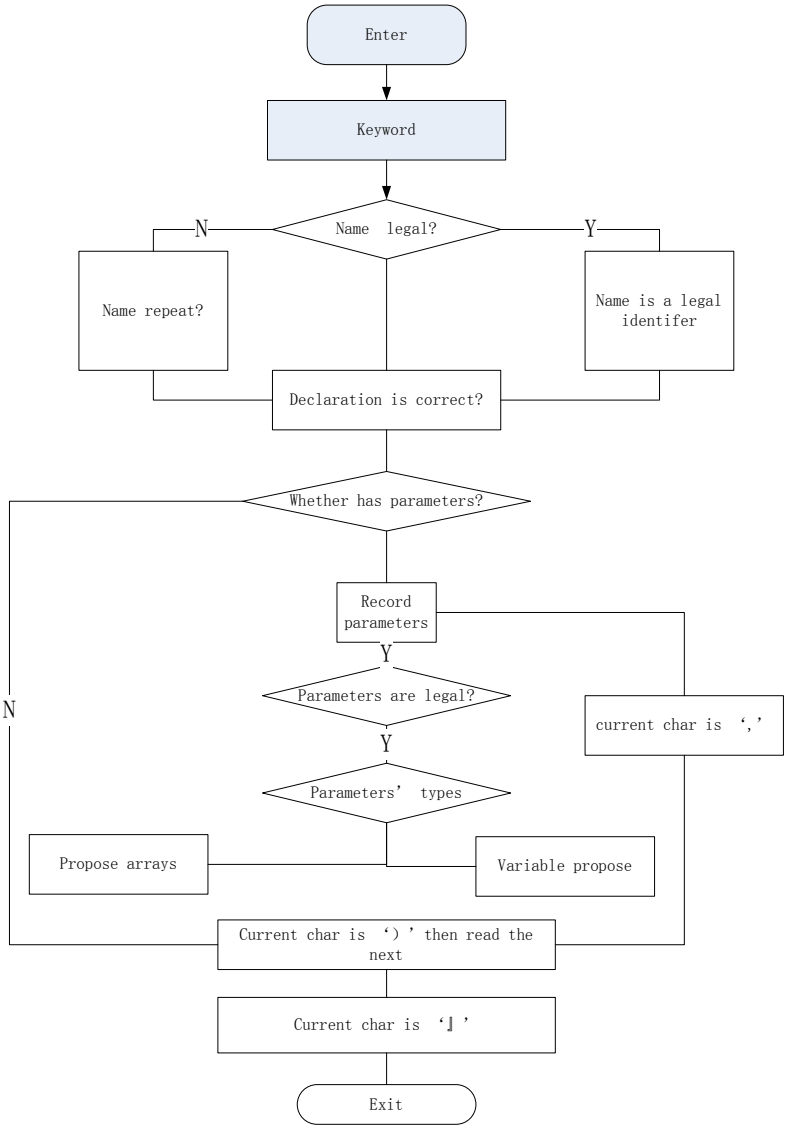

Fig.2. lexical analyzer of proposing function

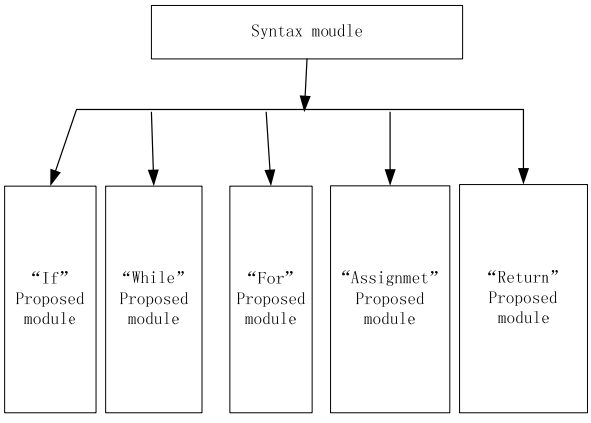

Fig.3. syntax analyzer

Fig.4. automatic construction

Conclusion

In this paper, we introduced a detailed complementation based on our former work including two methods. Since the manual method requires an explicit recognition of each step during compiling, it can lead to a deep study. Since shortcomings in the tools, students can improve the learning by making the compiling more efficient. The tools we changed or introduced for compiler construction gave them more freedom to experiment, more leisure to understand the core issues, more opportunities to find their mistakes by themselves. They also helped us automate the part. Our future work is to complement a compiler with tools mentioned above.

Acknowledgement

This work is supported by the Teaching Project of Shanghai Second Polytechnic University named “compiling technology” and Academic Affairs Division of Shanghai Second Polytechnic University and special funds of 085.

References

[1] Yang Dongyong. Return to Engineering: Education Reform to foster Applied Innovative Software Talents. Proceedings of 2009 4th International Conference on Computer Science & Education. 2009

[2] Wu Qidi. Reform and Development of China Higher Engineering Education[J]. China Higher Education Evaluation. 2007(4): 3- 7

[3] Zhang Chengcheng, Guo Shiying. A survey on the outline of National Long-Term Education Reform and Development Plan. Communication Software and Networks (ICCSN). 2011

[4] Dong, Hong, Zhang, ChuanYin. Discussions on the reform of higher vocational education from training the IT students’ occupational ability. E-Business and E -Government (ICEE). 2011

[5] Ming Xiao, Guang Hu. Research on Reform of Computer Elementary Education in University. Education Technology and Computer Science (ETCS). 2010

[6] Zheng Xiaojuan, Jin Ying. Engineering Reform of Compiler Principles Course. 2008 International Conference on Computer Science and Software Engineering. 2008

[7] Wang Na, Zhang ShiMing. Construction of compiler technology course in application-based university. 2013 1st International Conference on Education Technology and Information System. 2013

[8] Francisco Ortin, Daniel Zapico, and Juan Manuel Cueva. Design Patterns for Teaching Type Checking in a Compiler Construction Course. IEEE TRANSACTIONS ON EDUCATION, VOL.50, NO.3. AUGUST 2007