Error Log Analysis for Improving Educational Materials in C Programming Language Courses

Xinyu FU[a]*, Chengjiu YIN[b] , Atsushi SHIMADA[b], Hiroaki OGATA[b]
[a]*Department of Faculty of Information Science and Electrical Engineering, Kyushu University, Japan*
[b]*Faculty of Arts and Science, Kyushu University, Japan*
*fu.xinyu.277@s.kyushu-u.ac.jp

Abstract: Many universities choose the C programming language (C) as the first programming language to teach to students. As novice programmers, students frequently make simple mistakes such as syntax and typographical errors. Students often find it difficult to locate these errors, as students are not yet thoroughly familiar with C's syntax. Usually educational materials are very useful tools for students to locate errors and find solutions. This study aims to facilitate teaching and learning of C. We propose a system that undergraduate novice programmers may use to easily locate syntax errors in C and get recommendations from educational materials. We analyze error logs of programming and reading logs of educational materials, with the learning by doing mode (learning-practicing-reflection) to discuss key findings and their implications for programming education.

Keywords: C programming, educational materials, learning by doing, learning analysis

1. Introduction

Novice programmers typically do not understand C's syntax very well. Students learning C frequently make simple errors, such as typographical errors or careless use of syntax. Though these errors are simple, novices typically find their resolution difficult. Novices may struggle to locate the cause of errors, and may find the nature of the error obscure (Fu et al., 2015).

Robins and Rountree (2003) reviewed and discussed teaching and learning of programming. Their study examined novices, and discussed educational materials and showed that students considered programming courses difficult. However, most recent research examines improving learning of programming styles. Extant research examining effective education of novices is limited.

It is therefore necessary to facilitate the teaching and learning of C. C is the first programming language taught to students at our university, as a sound understanding of C is very helpful to learning other programming languages. Sharan (1980) showed that most traditional programming instructions usually focus on syntax, logics through lecturing and take place in classroom. Ala-Mutka (2004) suggested that "technical tools and visualizations are simply learning aids and materials. Teachers must thoroughly design their instructional approach to the issues in the course, and how the aiding materials are incorporated into education."

In Kyushu University, we used Booklooer-a e-book reader for digital teaching materials, we can collected logs during class (Yin et al., 2015).Teaching materials has been uploaded to BookLooper, students can use BookLooper anytime. We are able to gather materials reading logs from BookLooper's server.

We collected logs from students learning programming, and we had classified error types using error logs from undergraduate novice programmers (Fu et al., 2015). By using the research results, in this paper, we proposed a learning support system through using programming error logs and Booklooper reading logs. The research in this paper will mainly relevant to four points:

- Supporting students to locate students' errors and recommend related teach materials.
- Feedback the most common visited pages and related common error messages to teachers.
- More accurately to locate error by associated with error messages in the same error logs.
- Find the typical programming which contains same error messages, share these kinds of programming to students using in testing and self-checking.

The remainder of this text is structured as follows: section 2 provides a review of relevant programming and computing education literature. Section 3 provides a description of our study. Finally, we discuss our findings' implications, and indicate scope for further research.

2. Related Research

Burton (1998) suggested that teachers should keep in mind the principle distinct from the following modalities "what actually gets taught; what we think is getting taught; what we feel we'd like to teach; what would actually make a difference" in teaching Java. However this could equally apply to any kinds of educational situation. Of course for teaching C, teachers also should keep in mind of these problems.

Thus, extant research predominantly examines cooperative programming and self-education systems. Hwang et al. (2012) discussed cooperative learning of ASP.NET using the WPASC (Web-based programming assisted system for cooperation). The research they made found that cooperative programming style is useful for many students. We consider that if the WPASC system does not effectively manage cooperative programming, less able students may not do their best to resolve errors in their programs.

Nagao and Ishii (2003) proposed an agent support system for C. This system is also a type of cooperative programming: students may share knowledge and error resolutions through agent software. Park et al. (2015) analyzed HTML and CSS syntax errors in a web-development course. They examined the JavaScript programming language, and used the open HTML editor system, to analyze difficulties that novices experienced in learning HTML and CSS' syntax.

Problems in learning and teaching programming also be discussed (Robins at el., 2003; Ala-Mutka, 2004). However limit research are focus on the beginning stage or education materials. Chang et al. (2000) discussed the strategy approach of programming learning for beginners. Some research on programming tutors are listed in their research. However, it is not easy for beginners to initiate a program out of the blue, and that will make students lower their learning motive. How to make students faster and better understand the basic knowledge is really important.

Research examining initial education in the C language is limited. In our study, we propose a system that may facilitate students' understanding of why errors may be made, as well as how to resolve them. Further, we aim to analyze the logs between error logs and BookLooper reading logs to perfect the educational materials.

3. System Design and Methods

3.1 Environment and Data Source

Figure 1 illustrates our system's architecture. Students may connect to the server through terminal software TERA-TERM using his or her student ID and password. Students are not required to install any programming software; the compile software GCC (GNU Compiler Collection) has already installed in the server. Students can get themselves' workspace by using themselves account. All reports of GCC compiling will be stored in directory "./log". Students' compiling logs will be stored in the directory which named by student ID. We are able to gather logs from the server through SFTP.

In our previous work, we had used 53505 errors made a simple analysis on C error logs among novices. These logs are from 909 students attending a C programming course; 164 from autumn 2014, and 745 from spring 2015. All students did not have knowledge on programming.

In addition, the educational materials are uploaded to BookLooper with different topics. To better survey novices' syntax errors, we analyzed error messages with respect to the course schedule. Table 1 provides the course schedule, with topics and assessments organized by week. Students need to login BookLooper using student ID and password during classes, and download the educational materials to acquire knowledge. We are able to gather the BookLooper reading logs from BookLooper's server. Booklooper reading logs can refer to (Yin et al., 2015).

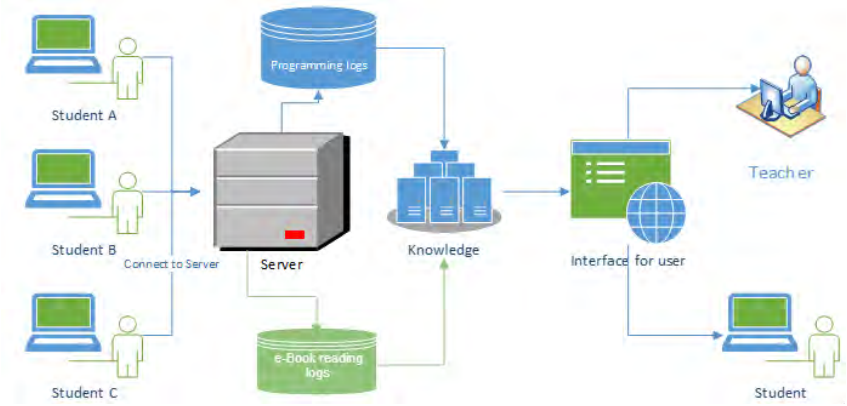

Figure1. The architecture of our system

3.2 Learning by doing in Course

Every C programming language course has 90 minutes in our college. Teachers will give more time for students to do the programming exercises. Figure 2 illustrates the structure of learning by doing in course. Teachers usually begin the course with about 15 minutes of instruction using the teaching materials in BookLooper. The remaining time will be used by students to finish 6 exercises. During practicing time, many programming errors will be made among students. Once the errors occurred, some students will ask TA (teaching assistants) or teacher for help. TA is not enough to follow every student (Nagao and Ishii 2003). Most students will reflect the knowledge in BookLooper. But sometimes knowledge introduced in educational materials on BookLooper may not so clear. We are thinking about how to make the error logs and BookLooper reading logs more useful and how to make the educational materials more specific. Problems like how to make the logs more useful, how to make the error types more accurately and how to reuse the error programming will be discuss in the following sections.

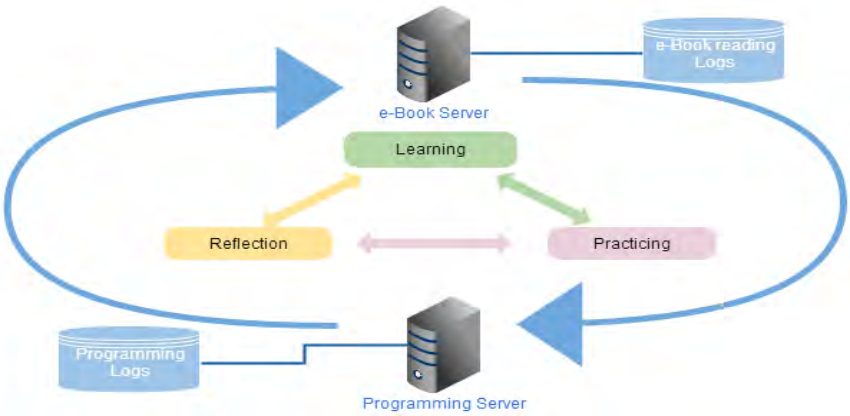

Figure 2. Illustrates the structure of practice learning in course

3.3 How to make logs more useful

The purpose of this research is to make C more easily for learning and make educational materials more effective on helping teaching C. Educational materials are very important reference material for novices during courses. We analyze educational materials using two kinds of logs: C programming error logs and BookLooper reading logs. Through the analyzing results to find which part of educational materials are not clearly clarifying and what contents need to be added.

In our previous work, we had already analyzed the most common error types of each course topic. For the next work, we propose to combine the analysis results of error types with the reading logs of BookLooper to analyze of the following problems. One is once one type error happened, which page is most visit for helping among all the students. The other one is whether the material on that page can help students to solve the problems. If it can't, what kinds of materials need to be added.

3.3.1 Scenario

Through using our system, when one error occurs, system will analyze the most visited pages of BookLooper to judge related materials. For example, when the type of misuse of switch statement error happened, the most happened error type in topic 6 which is expect the most common error types like missing semicolon: Multiple If-else and Switch-case. One program example is showed in figure 3. In this example, there have the errors in case statement, where have already changed to Bold.

By using the system mentioned in this paper, we find during the time of error happened page 9 of topic 6 are frequently visited. Figure 4 shows the page 9 on BookLooper. It is about very simple explanation on syntax of switch-case. In page 9 we notice that no special instructions are explained for the statement that after case should use ":", not ";". For novices, it is very easy to think that ";" should be used after every line, but in this case, it should be different. If teachers make an ambiguous interpretation for this syntax, it will make students make errors.

As a result, we will make a feedback on this to teachers. Send some typical error programs and the pages associated with the errors to teachers. Use the feedback teachers will know the explanation on which part is not enough. It is easy for teachers to make a decision on what should be added into the materials and which part should be explain more clearly in class. On the other hand, these kinds of often visit pages also will be used for students. As most of the pages is useful for modifying errors, we plan to recommend these pages to students in our future work. When they meet the same errors, the pages will be presented regarded as reference in student's homepage on our web system.

```
#include<stdio.h>
int main(){
int n;
printf("順位を入力してください¥n");
scanf("%d",&n);
switch (n){
case 1;
printf("優勝者には、100 万円の賞金です。\n");
break;
case 2;
  printf("2 位には、10 万円の賞金です。\n");
break;
default;
printf("記念品をどうぞ。\n");
}
return 0;}
```

Figure 3. Program with misuse of switch statement error

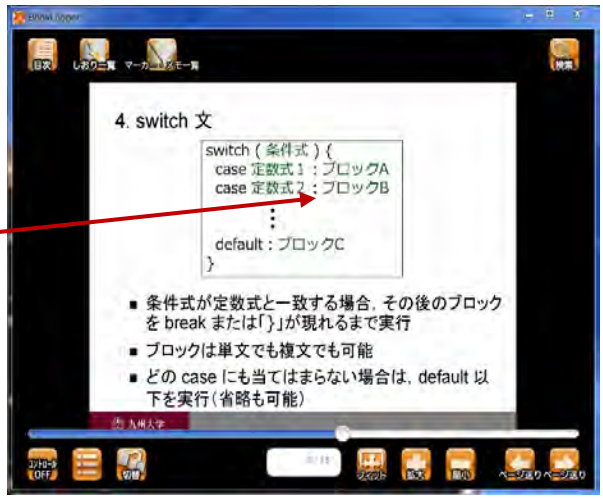

Figure 4. Details of topic 6 in page 9 on BookLooper

3.4 How to make error types more accurately

In our previous work, we had already classified the error logs into 26 types, the types will be added with the error logs added. During the process of analysis we found that in one program, usually will reports many error messages. We estimated every error message and feedback to students and teaches in our previous work. However, we found some error messages are depend on the early error message, the following part error messages are not so important. It is meaningless to recommend solution for these kinds of messages. Figure 5 give a simple example of this kind of program and the error messages.

Form the example, we notice that error: 'c' undeclared (first use in this function) is caused by missing semicolon of line 4. We just need to tell novice that the fundamental error of this program is missing semicolon in line 4. Based on this findings, we propose to combine error messages to make more accurately on classifying error logs. For example, error message "expected '=', ',', ';', 'asm' or '__attribute__' before" is happened before "undeclared (first use in this function)" in one program, it is usually because of the error "expected '=', ',', ';', 'asm' or '__attribute__' before", we will make a

model to indicate the root cause is from message “expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before”. We hope to make a more accurate way that can help novice locate the error more accurately.

```
#include<stdio.h>
int main(int argc, const char * argv[]){
   int a,b,c
   a=5;b=3;
   c=a+b;
   printf()"%d\n", c);
   return 0;
```

```
$ gcc example2.c
example2.c: In function ‘main’:
example2.c:3: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before ‘a’
example2.c:5: error: ‘c’ undeclared (first use in this function)
example2.c:5: error: (Each undeclared identifier is reported only once
example2.c:5: error: for each function it appears in.)
example2.c:6: error: too few arguments to function ‘printf’
example2.c:6: error: expected ‘;’ before string constant
example2.c:6: error: expected statement before ‘)’ token
example2.c:7: error: expected declaration or statement at
```

.

Figure 5. An example of error program with the error message

3.5 How to reuse error programming

Programming contains errors also very valuable resources. In this paper we also consider how to reuse error programs from novices. We collect students programs, and analyze error logs. We found some errors are common errors among students. We propose to change these programs to exercises and send to students at last of every course or in final test. We consider to use this programs on checking whether students really understand the syntax. Moreover we hope use these programs to make a self-checking for students. It is usually easier for students to remember knowledge through reflection.

4. Conclusions and Future Work

In this paper, we outline four main tasks on effectively assist learning of C. Our results may effectively perfect educational materials and help students to locate the cause of errors more accurately. Further, we will recommend the educational materials to students accurately, and help students check learning conditions using themselves’ programs. In our next work, we intend to discuss on how to make the error classify model accurately. We plan to finish our system quickly, and deploy it in a C programming course, in order to verify its usefulness. In addition, we will consider analyzing various methods such as social network analysis and visualization of graph theory (Mouri et al., 2014; 2015).

Finally, as students in C programming courses are a mixture of computing majors and general education students, we wish to analyze potential differences in rates of error types between majors, and adapt course teaching accordingly to each major. We intend to implement an online system for all novices studying C programming, and collect and examine more error messages and error types. It is our hope that our system may serve a greater number of learners, and be useful in more complex programs.

Acknowledgement

This research work was supported by the Grant-in-Aid for Scientific Research No. 25282059, No. 26560122, No. 25540091, and No. 26350319 from the Ministry of Education, Science, Sports, and Culture in Japan and "Research and Development on Fundamental and Utilization Technologies for Social Big Data" (178A03), the Commissioned Research of the National Institute of Information and Communications Technology (NICT), Japan.

References

Ala-Mutka, K. (2004). Problems in Learning and Teaching Programming - a literature study for developing visualizations in the Codewitz-Minerva project. *Codewitz Needs Analysis*. Literature Study.

Burton, P. (1998). Kinds of language, kinds of learning. *ACM SIGPLAN Notices*, 33, 53-61.

Chang, K.-E., Chiao, B.-C., Chen, S.-W., Hsiao, R.-S. (2000). A Programming Learning System for Beginners — A Completion Strategy Approach. *IEEE Transactions on Education*, Vol. 43, No. 2. , 211-220.

Fu, X.-Y., Yin, C-J., Shimada, A., Ogata, H. (2015). Error Log Analysis in C Programming Language Courses. *Proc. International Conference of Computers on Education, 2015*, in print.

Hwang, W.-Y., Shadiev, R., Wang, C.-Y., & Huang, Z.-H. (2012). A pilot study of cooperative programming learning behavior and its relationship with students' learning performance. *Computers & Education*, 1267-1281.

Mouri, K., Ogata, H., Uosaki, N. (2015) Ubiquitous learning analytics in the context of real-world language learning, *Proceedings of LAK15*, 378-382

Mouri, K., Ogata, H., Uosaki, N., Liu, S. (2014) Visualization for Analyzing Ubiquitous Learning Logs, *Proceedings of the 22nd International Conference on Computers in Education (ICCE 2014),* 461-470.

Nagao, K., Ishii, N. (2003). Evaluation of Learning Support System for Agent-Based C Programming. *Knowledge-Based Intelligent Information and Engineering Systems Lecture Notes in Computer Science* ,Vol. 2774, 540-546

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, Vol. 13, No. 2, 137-172.

Sharan, S. (1980). Cooperative learning in small groups: recent methods and effects on achievement, attitudes, and ethnic relations. *Review of Educational Research*, 50(2), 241-271.

Thomas, H. P., Dorn, B., & Forte, A. (2015). An analysis of HTML and CSS syntax errors in a web development course. *ACM Trans. Comput. Educ.* Vol. 15, No. 1, 4:1-4:21.

Yin, C-J., Okubo, F., Shimada, A., OI, M., Hirokawa, S., & Ogata, H. (2015). "Identifying and Analyzing the Learning Behaviors of Students using e-Books", *Proc. International Conference of Computers on Education, 2015*, in print.

Appendix table 1. Weekly Overview of the Course Schedule

Week	Topics	Assessments
1	Introduce to C Language	3 Exercises of Using "Printf" Statement
2	Variables	6 Exercises of Variables
3	Functions	6 Exercises of Using "Scanf" Statement and Operator Symbol like "+", "-", "*", "++".
4	Mathematical Functions	6 Exercises of Mathematical Functions
5	Decision Making Structures If-else	6 Exercises of If-else
6	Multiple If-else and Switch-case	6 Exercises of Multiple If-else and Switch-case
7	For Loop	6 Exercises of For Loop
8	Array	6 Exercises of Array
9	Multi-dimensional Array	6 Exercises of Multi-dimensional Array
10	Multiple For Loop	6 Exercises of Multiple For Loop
11	While--Do-while Loop	6 Exercises of While--Do-while Loop
12	String Functions	6 Exercises of String Functions
13	User-defined Functions	6 Exercises of User-defined Functions
14	File I/O	6 Exercises of File I/O
15	Programming Practise	1 Exercises of Final Test of Semester