Real-time Learning Analytics for C Programming Language Courses

Xinyu Fu
Graduate School of Information Science and Electrical Engineering
Kyushu University
Japan
fx0207@gmail.com

Atsushi Shimada
Faculty of Arts and Science
Kyushu University
Japan
atsushi@artsci.kyushu-u.ac.jp

Hiroaki Ogata
Faculty of Arts and Science
Kyushu University
Japan
ogata@artsci.kyushu-u.ac.jp

Yuta Taniguchi
Faculty of Arts and Science
Kyushu University
Japan
taniguchi@artsci.kyushu-u.ac.jp

Daiki Suehiro
Faculty of Arts and Science
Kyushu University
Japan
suehiro@artsci.kyushu-u.ac.jp

ABSTRACT
Many universities choose the C programming language (C) as the first one they teach their students, early on in their program. However, students often consider programming courses difficult, and these courses often have among the highest dropout rates of computer science courses offered. It is therefore critical to provide more effective instruction to help students understand the syntax of C and prevent them losing interest in programming. In addition, homework and paper-based exams are still the main assessment methods in the majority of classrooms. It is difficult for teachers to grasp students’ learning situation due to the large amount of evaluation work. To facilitate teaching and learning of C, in this article we propose a system—LAPLE (Learning Analytics in Programming Language Education)—that provides a learning dashboard to capture the behavior of students in the classroom and identify the different difficulties faced by different students looking at different knowledge. With LAPLE, teachers may better grasp students’ learning situation in real time and better improve educational materials using analysis results. For their part, novice undergraduate programmers may use LAPLE to locate syntax errors in C and get recommendations from educational materials on how to fix them.

CCS Concepts
• Social and professional topics → Professional topics → Computing education → Computing education programs → Computer science education → CS1.

Keywords
C programming; programming education; learning analytics; information visualization; learning dashboard

1. INTRODUCTION
Programming is a very useful skill and many universities choose C as the first programming language to teach to students. However, novice programmers typically do not understand C’s syntax very well, and frequently make simple errors, such as typographical errors or careless use of syntax. Though these errors are simple, novices typically find their identification and resolution difficult. That is, they may struggle to locate the cause of the errors, or may find the nature of the error obscure [6]. In addition, as teachers are mostly competent programmers, it is easy to omit to discuss simple errors while explaining course content. It is, therefore, necessary to devote some attention to how best to address these issues to facilitate the teaching and learning of C.

Previous research on teaching and learning of programming is reviewed and discussed in [16, 19]. Those studies reviewed some research on examined novices and discussed some different teaching methods and indicated that students considered programming courses difficult. Most traditional programming instruction focuses on syntax and logic and is delivered through lectures in the classroom [10, 20], but “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” [1]. To effectively facilitate teaching and learning of C, it is important to identify the different difficulties faced by different students with different background grappling with different target knowledge, in a way that reflects their (different) behaviors in the classroom.
In our last study [6], we collected compiling logs from novice students learning programming and classified error types on their basis. This revealed that a large number of students didn’t understand the basic concepts of the syntax very well, and some didn’t know how to apply conceptual knowledge even after sitting in on related lectures. Another very interesting finding is that when some students received compiling errors, they simply tried to compile the program again and again, without any modification.

The main purpose of this article is to present a tool to help identify the weaknesses of novice programmers in order to improve teaching materials supporting C education in the classroom. The tool we propose, LAPLE (Learning Analytics in Programming Language Education), may be used by novice undergraduate programmers in the class. On the one hand, LAPLE can be used to locate syntax errors in C and get recommendations to address them derived from educational materials. On the other hand, teachers may be able to better grasp students’ learning situation using LAPLE. We plan to analyze programming error logs and class material reading logs, reflecting the learning by doing mode (learning-practicing-reflection; figure 1) adopted at Kyushu University, our institution, and on the basis of the analysis to discuss key findings and their implications for programming education.

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 own empirical study. Section 4 presents the results gathered by our learning dashboard. Finally, we discuss our findings’ implications, and sketch out scope for further research.

2. RELATED RESEARCH

2.1 Research on Programming Education

Many studies have been done with the intention of gathering knowledge to support people learning programming. Extant research predominantly examines cooperative programming and self-education systems [10, 12, 15 and 19]. The relationship between students’ cooperative programming behavior and their learning performance was investigated for cooperative learning of ASP.NET [10]. The results revealed that using WPASC (Web-based Programming Assisted System for Cooperation), students who are active in programming will conduct more activity. However, more work is still needed to increase learning motivation for students who have poor learning performance. An agent support system [12] for C provides the opportunity for students to share knowledge and conduct error resolution through agent software, as another type of cooperative programming. The results indicated that the agent system is useful for programming, but also that novices need more support on programming, for example, similar examples. In another study, HTML and CSS syntax errors were analyzed in a web-development course [15]. The authors examined students’ use of the JavaScript programming language and the open HTML editor system to analyze what difficulties these novices experienced in learning HTML and CSS syntax. The results revealed that syntax errors were made by almost all the students and usually persisted for a very long time in students’ programming efforts.

Certain systems have been suggested and provided to help make programming and debugging easier for students [3, 9]. However, these systems are much more useful for students who already have some programming knowledge; how to effectively support programming novices to avoid, correct, and understand syntax errors in the classroom is also very important.

Various methods have been suggested to support programming education in a way that reflects students’ needs. These include assessment of learning standards in programming courses by benchmarking a set of exam questions [21] and studying debugging behavior in order to better understand and address students debugging errors [11]. Monitoring students’ programming efforts in detail is very important for teachers in programming education, who are then able to determine when students need guidance and what kind [7]. Teachers should keep in mind that around any principle or concept being taught orbit 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” [2].

However, with so much research done to understand novices’ programming behavior and make programming easier for them, research on supporting teaching and learning at the beginning in the classroom or on improving education materials is still limited. How to help students understand basic conceptual and technical knowledge faster and better is really important.

2.2 Supporting Education using Dashboard Applications

“A dashboard is a visual display of the most important information needed to achieve one or more objectives, consolidated and arranged on a single screen so the information can be monitored at a glance” [5]. In recent years, several dashboard applications have been developed to support learning and/or teaching [4, 8, 17 and 22]. Learning process are made easier and more comfortable by using such educational data. Advanced data analysis can provide immediate feedback on students’ engagement and performance in educational activities, and on that basis guide students to appropriate learning materials and help teachers discover where students’ weaknesses are [4, 17]. A semantic visual analytics tool for programming courses (EduAnalysis) is designed to provide teachers with immediate feedback on students’ exams, allowing them to produce more balanced exams [8]. Some learning analytics dashboards and data visualizations are available in Moodle, a common learning management system, for analyzing the relationship between motivation and students’ success on exams; these have had positive, enthusiastic feedback [13-14, 17]. Here, we propose a learning dashboard system that may facilitate students’ understanding of C as well as supporting teachers. Further, we aim to analyze the correspondence between error logs and BookLooper reading logs and apply the results to improve educational materials.
3. DATA ANALYSIS AND VISUALIZATION

3.1 Environment and Data Source

Kyushu University employs a single-platform learning system (Mitsuba, M2B) constructed using various tools: Moodle; Mahara, an e-portfolio system; and BookLooper, an e-book system [13]. Our present research mainly relies on Moodle and BookLooper. As LAPLE is a plug-in to Moodle, all teachers and students may use it in class. Teaching/learning materials are uploaded to BookLooper, which students can use; we are then able to gather reading logs from BookLooper’s server.

The C programming language course at our university is organized into 90-minute sessions. It begins with about 20 minutes on imparting C’s knowledge by teachers using the teaching materials in BookLooper. In the remaining course time, students need to finish 6 assignments (2 difficult, 2 average, and 2 simple). The teaching approach used is one of “learning by doing (figure 1)”; if students make errors or encounter problems that they cannot solve during practice time, they can ask the TA (teaching assistants) or the teacher for help, or can investigate themselves using BookLooper. Using this approach, we can collect and review students’ learning log information in real time in the classroom.

![Figure 2. The construction of the LAPLE system.](image)

Figure 2 illustrates the construction of our LAPLE system. In the initial lecture, the teacher and students log in to Moodle, and the teacher introduces the materials on BookLooper which is already linked into Moodle. Then, the students work on the exercises, as described. To help students work out their compiling programs, our university provides students with a common server they can connect to through terminal software TERA-TERM using their student ID and password. Students are not required to install any programming software; the compiled software GCC (GNU Compiler Collection) is already installed on the server. Students can access a workspace by using their account. We gather logs from the server through SFTP (SSH File Transfer Protocol) in real time, analyze the content, and provide feedback to users. From October 2014 to June 2016, we collected 989,560 compiling error messages from 1975 students. The detailed data are shown in table 1.

To better assess novices’ syntax errors, we categorized error messages in terms of the course schedule (appendix table 1). Students need to log in to BookLooper during classes to download educational materials and consult them to acquire needed knowledge. We are able to gather BookLooper reading logs from BookLooper’s server. For basic information on BookLooper reading logs, refer to [23].

<table>
<thead>
<tr>
<th>Semester</th>
<th>Student Code</th>
<th>Error Logs</th>
<th>Source Code</th>
<th>Courses</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014 2nd</td>
<td>164</td>
<td>11,581</td>
<td>17,642</td>
<td>2</td>
</tr>
<tr>
<td>2015 1st</td>
<td>745</td>
<td>607,076</td>
<td>153,808</td>
<td>17</td>
</tr>
<tr>
<td>2015 2nd</td>
<td>404</td>
<td>265,602</td>
<td>104,943</td>
<td>6</td>
</tr>
<tr>
<td>2016 1st</td>
<td>662</td>
<td>105,301</td>
<td>19,073</td>
<td>20</td>
</tr>
<tr>
<td>All</td>
<td>1,975</td>
<td>989,560</td>
<td>295,466</td>
<td>45</td>
</tr>
</tbody>
</table>

3.2 System Design and Methods

The advance of internet-capable consumer technologies has led to many learning support systems being made available online for people learning programming [10, 12, 15 and 19]. However, most students still acquire programming knowledge from traditional classroom education, and teachers usually determine whether or not students have acquired knowledge through the reports of homework and exams. On the one hand, cheating remains a big issue; on the other, it is a lot of work for teachers to check reports evaluate the knowledge evidenced therein. Further, without accurate feedback, making meaningful teaching materials is very time consuming.

To support teaching and make the learning process easier, we propose the web system LAPLE for students and teachers. Users may use their account to browse the analyzed results. Base on the traditional C programming learning path, we proposed a new design with LAPLE for supporting C programming education (figure 3). The detailed functions are described in the following subsections.

![Figure 3. System design.](image)

3.2.1 How does LAPLE support teaching?

The LAPLE system provides real-time feedback. Real-time analysis allows teachers to grasp student programmers’ learning styles and problems [18], with proven benefits for the students, who can get targeted guidance as faster than otherwise. We set LAPLE to collect and analyze logs every 5 minutes to yield real-time visualization feedback.
Students who try to compile two or more programs but do not address all errors are classified into level C; those who work on only one program and try to compile it for more than 20 minutes, into level D. Students in levels C and D are the students who have troubles with programming but are still trying hard to “get it.” These students can be considered active students. Those who work on only one program for less than 20 minutes but give up are classified into level E, these students also can be considered as inactive students. Teachers and TAs should provide one-on-one instruction to students in levels C, D, and E.

### 3.2.1.1 Discover students’ weaknesses and critical knowledge points for teaching

We collected and analyzed students’ compiling logs with the intention of identifying high-frequency errors and those which may need more time to rectify. Besides allowing teachers to give students timely, targeted feedback, the results can help convey effectively and modify class materials accordingly.

Analysis was critical during the early stages of the system’s preparation. Since at the start of the analysis we had already collected 11,581 different error messages (62,812 by May 2016), it was of course impractical to try to explain or provide resolutions for each one individually. Instead, to understand the syntax errors that novices commonly make and on this basis to effectively improve outcomes in C programming education, we classified these errors into 26 types; these initial findings have already been discussed in previous work [6]. As more and more students take the course and compile their programs, error messages are updated and new ones may appear. At present, we have identified 36 (see appendix table 2).

<table>
<thead>
<tr>
<th>Level</th>
<th>Evaluation Method</th>
<th>Evaluation of internal order in the same level</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Complete at least 4 programs (all 6 programs), and no remaining errors.</td>
<td>[ X = \frac{\text{Time of Programming}}{\text{programs}} ] The X is smaller, the student is more excellent. (Some students can finish a lot of programs in a short time.)</td>
</tr>
<tr>
<td>B</td>
<td>Complete at least 2 programs (all 6 programs), and no remaining errors.</td>
<td>[ X = \frac{\text{Time of Programming}}{\text{programs}} ] The X is smaller, the student is more excellent. (Some students can finish a lot of programs in a short time.)</td>
</tr>
<tr>
<td>C</td>
<td>Complete at least 2 programs (all 6 programs), finally, there are still some errors.</td>
<td>[ X = \frac{\text{Time of Programming}}{\text{programs}} \ast \text{remaining errors} ] The X is bigger, the student needs more help. (Some students need more time to finish programs and cannot fix the problems in programs.)</td>
</tr>
<tr>
<td>D</td>
<td>Complete less than 2 programs (all 6 programs), but try to program more than 20 minutes.</td>
<td>[ X = \frac{\text{Time of Programming}}{\text{programs} + 1} \ast \text{remaining errors} ] The X is bigger, the student is work harder. This part of students needs more help about simple knowledge in programming.</td>
</tr>
<tr>
<td>E</td>
<td>Complete less than 2 programs (all 6 programs), and the time of trying to programming is less than 20 minutes.</td>
<td>[ X = \frac{\text{Time of Programming}}{\text{programs} + 1} \ast \text{remaining errors} ] The X is bigger, the student is work harder. This part of students do not study hard, and enthusiasm is very low.</td>
</tr>
</tbody>
</table>

3.2.1.2 Support teachers to grasp students learning situation in real time

By grouping the students into different knowledge levels, we detect which students are in trouble so as to provide support immediately.

We group the students by the time they spend on programming, how many compiling errors they fix, and how many exercises they finish in class. We set 5 knowledge levels on the basis outlined in table 2. Students in levels A and B can be considered outstanding students; teachers do not need to pay close attention to supporting them fix their programs, but can compose and provide them with more complex exercises to help them develop and produce the best work they can. Level A students complete more exercises with no compiling errors left, and can be considered excellent students. Level B students have good performance in programming, and although they can’t finish all the exercises, they are able to modify the errors that occur. Students in levels C, D, and E are the ones that need more help and the ones we especially want to identify.
here, colored in red. Our analysis results showed that the most visited page at the time was page 7 of topic 2.

As most-visited pages have shown themselves to be useful for addressing errors, LAPLE can then recommend these pages to students and link to them on students’ homepage as resources. Identifying the most visited pages can also help teachers discover which parts of material students are having trouble with, what should be explained more clearly in class, or whether they need to improve the content of some material.

4. RESULTS OF DATA VISUALIZATION

We provide LAPLE as a Moodle plugin so that all the students and teachers who attend the course are able to get feedback from our analysis. The matter of supporting reflection on learning is still in process. In this section, we will mainly discuss the lessons that can be learned from the error data and the feedback provided to teachers and TAs to support teaching through the learning dashboard (seen in figure 5, details can be seen in figures 6-11). The dashboard contains two main parts: the left part indicates the error distribution among students, and the right part indicates students’ learning situation. The details are described in the following subsections.

4.1 Learning Dashboard Data on Error Types

The error data are mainly used to discover students’ weaknesses and critical knowledge points for helping them acquire the material.

The bar chart is used to show the error distribution by course topic, based on the log data collected so far. Teachers may be able to provide more targeted explanations of C programming topics with which students have difficulty based on the results. For example, figure 6 clearly illustrates that errors of types 27, 26, 11, 20, 23, 4, 24, 8, 5, 28, 10, and 9 are most encountered by students in topic 4. Among these, type 26, 20 and 10 are mostly caused by missing semicolons, while type 23 is caused by mistyping of the standard library of C. From our experience, these kinds of errors do not decrease as the course progresses; teachers should, therefore, emphasize these error types and how to address them throughout the course. Types 8 and 9 involve incorrect use of mathematical functions in relation to content in topic 4; the high frequency rates thus indicate that teachers should spend more time on explaining mathematical functions, especially pow functions, during class coverage of this topic. Further, these results are useful for improving teaching materials, so that different knowledge points can be explained more concretely.

Further, as the results remain visible until the next class, teachers may summarily explain simple errors made on the previous class’s topic. Additionally, the results can be referred to later on to optimize C education materials.

4.2 Learning Dashboard for Grasping Students’ Learning Situation

The charts shown in the figures help teachers grasp students’ learning situation in real time. The intention is to identify students as “outstanding,” “active,” or “struggling.” The outstanding students are those who can correctly complete most exercises by themselves; active students are those who keep trying and
modifying their code although they meet many compiling errors; and inactive students are those who are frustrated by errors and give up or do not spend enough time on programming. It is possible to provide targeted guidance to students through this grouping.

Figure 8 use a heat-map chart to indicate the students’ activity. We collect logs every 5 minutes and update the analyzed results to Moodle so that teachers can get feedback almost in real time. The numbers on the chart show how many times students try to compile programs (with larger numbers in darker color). With this heat-map chart, we can easily detect the activity and inactivity of students. For example, the student highlighted in green on the image is active, since he keeps trying to compile his program over the whole course of the class. In contrast, the student in blue only tries to compile 4 times within one 10-minute period, across the whole class; this student is inactive. In traditional evaluation in the C course, students’ reports of these exercises are usually a big part of their course mark. Some students do not really try to engage with programming, and instead submit reports copied from other students. We suggest that teachers can conduct a fairer evaluation of the reports with reference to the heat-map chart.

Figure 8 indicates all students’ activity levels; however, it shows only activity, not which students get outstanding results, and it is also not easy to see what kind of help a given student needs. The line-point chart in figure 9, developed using the method described in section 3.2 above, allows grouping by academic achievement, with levels A and B (as described above) classified as outstanding, C and D as active, and E as inactive. In this chart, red circles mean that at that timestamp the student still has compiling errors that need to be fixed, and blue means no errors left. The size of the point shows the number of times trying to compile; once the new program begins to compile, the point will become small again. The lines help us see the progress of individual programs; when the line turns up, it means a new program is beginning to compile. These results are also updated every 5 minutes, so that teachers can see which students need help immediately (making which students are in levels C, D, and E the most important information). The teacher can easily locate the student by name on the chart and provide appropriate support.

Further some teachers will want to know not only the compiling situation of the students but also which type of program is really accruing difficulties. In LAPLE, teachers can get detailed information on one student, as shown in figure 10. We flag the different difficulty grades of exercises into four colors in descending order of difficulty: blue, orange, green and red. A-grade is colored by blue, which is the highest difficulty level. B-grade is colored by orange, which is the second-highest. C-grade is green, which is the part of most easily. Before doing this research, we named the exercises in the teaching materials so that we could
distinguish the programs from students’ compiling logs. Some students named their programs as a rule, however some did not. To analyze all the logs that we collected, we add a red part which shows those without the right rule, like “2-a-1.c”. From figure 10, teachers are able to see which level of exercises students are trying to do, how many times the student tries to compile, how long the student works on the program, and how many errors the student makes at different stages.

5. CONCLUSIONS
In this article, we proposed a C education support system—LAPLE—discussed how to support teachers to more effectively educate students in their C programming classes with the real-time learning dashboard. Our work makes the following three main contributions. First, the error type visualization, which was analyzed in our previous research [6], allows us to characterize students’ weaknesses in their understanding of C material. To our knowledge, our study is the first to offer analysis of novices’ coding errors in a C course, and use the results to suggest ways to optimize education materials. Second, we use real-time analysis to support immediate feedback, so that teachers are able to provide effective and timely explanation when they notice the students who are in trouble. Further, we suggested combining programming logs and e-book reading logs to allow automatic recommendation of relevant material. These approaches have in common that knowledge gleaned from students is used to benefit the students. We plan to finish this part of work soon, so that LAPLE can be applied as soon as possible to support students’ learning.

However, at least one limitation of the study should be reflected. Given that we currently assemble error logs on the basis of syntax error messages obtained from the compiler, the causes of the errors we receive are not known exactly. A new error message analysis model will need to be created that allows us to obtain more precise error causes and, accordingly, better support learning.

The LAPLE system will be deployed in C programming courses from November 2016 (autumn semester) to verify its usefulness. Finally, as students in C programming courses are a mixture of computing majors and others, we wish to investigate potential differences in rates of error types between majors and adapt course teaching accordingly. To do so, we intend to implement an online system for all novices studying C programming in order to collect and examine more error messages and error types. It is our hope that the LAPLE system may in that way serve a greater number of learners, and be useful in more complex programs.

6. APPENDIX TABLE

<table>
<thead>
<tr>
<th>Week</th>
<th>Topics</th>
<th>Assessments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduce to C Language</td>
<td>3 Exercises of Using “Printf” Statement</td>
</tr>
<tr>
<td>2</td>
<td>Variables</td>
<td>6 Exercises of Variables</td>
</tr>
<tr>
<td>3</td>
<td>Functions</td>
<td>6 Exercises of Using “scanf” Statement and Operator Symbol like “+”, “-”, “*”, “++”</td>
</tr>
<tr>
<td>4</td>
<td>Mathematical Functions</td>
<td>6 Exercises of Mathematical Functions</td>
</tr>
<tr>
<td>5</td>
<td>Decision Making Structures If-else</td>
<td>6 Exercises of If-else</td>
</tr>
<tr>
<td>6</td>
<td>Multiple If-else and Switch-case</td>
<td>6 Exercises of Multiple If-else and Switch-case</td>
</tr>
<tr>
<td>7</td>
<td>For Loop</td>
<td>6 Exercises of For Loop</td>
</tr>
<tr>
<td>8</td>
<td>Array</td>
<td>6 Exercises of Array</td>
</tr>
<tr>
<td>9</td>
<td>Multi-dimensional Array</td>
<td>6 Exercises of Multi-dimensional Array</td>
</tr>
<tr>
<td>10</td>
<td>Multiple For Loop</td>
<td>6 Exercises of Multiple For Loop</td>
</tr>
<tr>
<td>11</td>
<td>While--Do-while Loop</td>
<td>6 Exercises of While--Do-while Loop</td>
</tr>
</tbody>
</table>
Appendix Table 2. Error types

<table>
<thead>
<tr>
<th>Type number</th>
<th>Error Description</th>
<th>Type number</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>Unmatched data type</td>
<td>Type 19</td>
<td>Missing semicolon before “return”</td>
</tr>
<tr>
<td>Type 2</td>
<td>Re-declaration of variables</td>
<td>Type 20</td>
<td>Missing semicolon</td>
</tr>
<tr>
<td>Type 3</td>
<td>Mismatch of “{“ (particularly, “{” after main)</td>
<td>Type 21</td>
<td>“,” used after variables, not “;”</td>
</tr>
<tr>
<td>Type 4</td>
<td>Undeclared variables (particularly, mismatched symbols and mistyping of symbols)</td>
<td>Type 22</td>
<td>Missing semicolon or comma</td>
</tr>
<tr>
<td>Type 5</td>
<td>Syntax errors (invalid operand or invalid suffix)</td>
<td>Type 23</td>
<td>Mistyping of standard library</td>
</tr>
<tr>
<td>Type 6</td>
<td>Unmatched variable type for array</td>
<td>Type 24</td>
<td>Full-width characters are used</td>
</tr>
<tr>
<td>Type 7</td>
<td>Mistakes on array declaration</td>
<td>Type 25</td>
<td>Misuse of switch statement (with or without use of “break”)</td>
</tr>
<tr>
<td>Type 8</td>
<td>Misuse of pow function</td>
<td>Type 26</td>
<td>Undeclared variables, or “;” or “}” is missing in the previous row</td>
</tr>
<tr>
<td>Type 9</td>
<td>Misuse of mathematical functions</td>
<td>Type 27</td>
<td>Miss input symbol such as “;”, “&lt;=&gt;”, or the mismatch of “[” symbol</td>
</tr>
<tr>
<td>Type 10</td>
<td>Missing punctuation (e.g. “=?” or “,” or “;” or “asm” or “<em>attribute</em>”)—semicolons were most frequently missing</td>
<td>Type 28</td>
<td>Errors in definition of variables or the boundary of the main function is wrong</td>
</tr>
<tr>
<td>Type 11</td>
<td>Other error type</td>
<td>Type 29</td>
<td>Unmatched symbols such as “()”, “{” “}”</td>
</tr>
<tr>
<td>Type 12</td>
<td>Statement is out of main class</td>
<td>Type 30</td>
<td>Variables should define before the “for” loop</td>
</tr>
<tr>
<td>Type 13</td>
<td>Missing “}” at the end of code</td>
<td>Type 31</td>
<td>“()”, “{” “}” symbols are not matched in if-else statement</td>
</tr>
<tr>
<td>Type 14</td>
<td>Mismatch of “}”</td>
<td>Type 32</td>
<td>Parameters’ type is incorrect in the function</td>
</tr>
<tr>
<td>Type 15</td>
<td>Mismatch of quote marks, or mismatch of “&lt;=&gt;”</td>
<td>Type 33</td>
<td>Array size should be defined as a constant</td>
</tr>
<tr>
<td>Type 16</td>
<td>Mismatch of “{” missing semicolons or comma before “}”</td>
<td>Type 34</td>
<td>Miss input variable’s type</td>
</tr>
<tr>
<td>Type 17</td>
<td>Re-declaration of variable type</td>
<td>Type 35</td>
<td>Misuse of do-while statement</td>
</tr>
<tr>
<td>Type 18</td>
<td>Two main classes in one program</td>
<td>Type 36</td>
<td>Variable’s type in return statement is unmatched desired variables</td>
</tr>
</tbody>
</table>

7. ACKNOWLEDGMENTS
This research work was supported by the Grant-in-Aid for Scientific Research No. 26560122 and No. 16H06304 from the Ministry of Education, Science, Sports, and Culture of Japan and by “Research and Development on Fundamental and Utilization Technologies for Social Big Data” (178A03), Commissioned Research of the National Institute of Information and Communications Technology (NICT), Japan.

8. REFERENCES


