MAX/C ON SAKAI
A Web-based C-Programming Course

Souichirou Fujii, Kazunori Ohkubo
Graduate School of Science and Technology, Meiji University
1-1-1 Higashi-mita, Tama-ku, Kawasaki-shi, Kanagawa, Japan

Hisao Tamaki
School of Science and Technology, Meiji University, 1-1-1 Higashi-mita, Tama-ku, Kawasaki-shi, Kanagawa, Japan

Keywords: e-Learning, C programming, Algorithmic feedback.

Abstract: MAX/C is a web application for C programming education, which has been successfully used in class of a computer science department for more than 4 years. It features algorithmically generated feedbacks both for drill questions and for programming exercise submissions. For the latter, source programs submitted by the students are statically analyzed as well as executed and tested against test cases. The execution is performed by a C interpreter which is developed for this purpose and designed to facilitate the detection of potential run time errors and the collection of useful information from the execution. We describe this system, together with the ideas behind the system. We also describe the recent integration of MAX/C into Sakai, an open source e-Learning environment.

1 INTRODUCTION

1.1 MAX: Project Overview

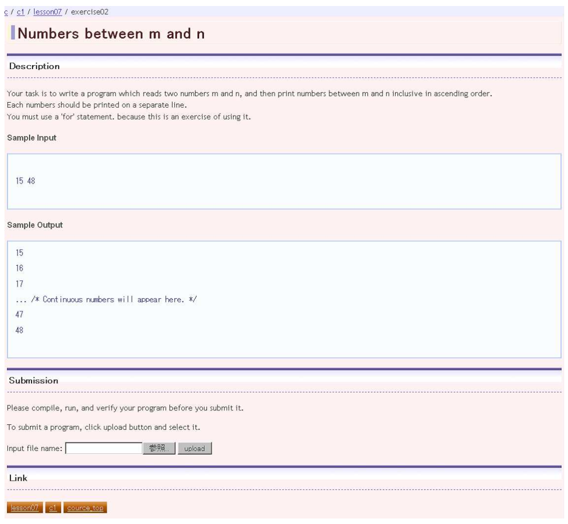
MAX (Massive Algorithmic eXerciser) is a project being undertaken at the computer science department of Meiji university, to which the authors belong. This project is motivated by the belief that skills and working knowledge in science and technology can only be achieved through a mass exercises. For those exercises to be effective, they must be interactive: an immediate and helpful feedback must be supplied for each action of the students. To meet this requirement, within the limitation of realistic teaching resources, an appropriate use of information technology is mandatory. Our approach is to let those feedbacks generated by algorithms that are designed individually for each problem domain or even for each exercise question or task, rather than by a small set of general principles or by artificial intelligence. Creating course materials along this line for each subject domain is a task that is enormously laborious but also is worthwhile, as successful outcomes can be used over and over again and across academic institutions.

The goal of this paper is to describe MAX/C, a web-based system for C programming education which is a product of a subproject of this project, and evaluate its effectiveness.

1.2 MAX/C: Overview

MAX/C is a web application for learning C programming, intended to be used primarily in class. Along the MAX philosophy stated in the previous subsection, it aims at letting each student learn through working on their own, rather than through listening or reading. Each of the two practice courses that use MAX/C consists of 14 3-hour classes, one given in each week. Accordingly, the contents of MAX/C is divided into two parts, each consisting of 13 lessons; an extra class is used for a confirmation test, which tries to confirm that each student has submitted the results of their own work. The first part introduces basic components of the language such as variables, assignments, *if*, *while*, and *for* statements, arrays, and functions. The second part focuses on more advanced topics such as structures, pointers, and recursive calls. A lecture-type course on the same subject runs in par-

Fujii S., Ohkubo K. and Tamaki H. (2010).
MAX/C ON SAKAI - A Web-based C-Programming Course.
In *Proceedings of the 2nd International Conference on Computer Supported Education*, pages 196-201
DOI: 10.5220/0002777501960201
Copyright © SciTePress


c / c1 / lesson07 / exercise02

Numbers between m and n

Description

Your task is to write a program which reads two numbers m and n, and then print numbers between m and n inclusive in ascending order.
Each numbers should be printed on a separate line.
You must use a 'for' statement. because this is an exercise of using it.

Sample Input

```
15 48
```

Sample Output

```
15
16
17
... /* Continuous numbers will appear here. */
47
48
```

Submission

Please compile, run, and verify your program before you submit it.

To submit a program, click upload button and select it.

Input file name:

Figure 1: A screen-shot of an exercise problem.

allel in a synchronized manner.

MAX/C provides both online questions, called drills, and offline tasks, called exercise problems, each lesson typically consisting of 10 to 20 drills and around 10 exercise problems.

Drills are further classified into yes/no or multiple choice questions and more interactive questions. A typical example of the latter is a trace question: a piece of C code, together with the current values of variables, is given and the student is asked to type in the changes of variable values that occur during the execution of the code. The initial values of the variables are generated randomly, so that the checking of the answers must be done algorithmically.

Most of the exercise problems are programming tasks. The input/output specification of the code to be written is described on the web page. Students use an editor on the local machine to write a program to fulfill the requirement, debug it and, when they believe it is correct, upload it through the same MAX/C page where the exercise problem is posed. Figure 1 shows a screen-shot of a page describing an exercise problem, which asks the student to write a program that prints consecutive numbers between two given numbers using a *for* statement. How MAX/C deals with the submissions to exercise problems is described in Section2 in detail.

Presently, the authors of drills and exercise problems, except for simple yes/no or multiple choice questions, have to provide, together with a text describing the questions, a Java code that checks the answer and generates feedbacks. This is an essential part of the MAX approach and therefore is inevitable. Nonetheless, we strongly feel the need of reducing the burden of the authors. We briefly touch on our current effort in this direction in Section5.

The new features to support course managements and communication among teachers, teaching assistants, and students are described in Section3 in details.

1.3 Related Work

Programming is a challenging target for computer supported education and there has naturally been a great amount of work on developing so-called intelligent programming tutors (Anderson and Skwarecki, 1986; Arnow and Barshay, 1999; Hong, 2004; Khudobakhshov, 2009; Lehtonen, 2005; Odekirk-Hash and Zachary, 2001; Pillay, 2003; Sykes and Franek, 2004; Xu and Chee, 2003; Truong et al., 2002). There have also been reports on successful uses of such systems in class (Pillay, 2003) and even some commercial systems that have evolved out of these research efforts, such as CodeLab (Turing's Craft, 2002; Arnow and Barshay, 1999). Yet most programming courses in colleges and universities are still taught solely by human instructors and the web is full of online programming courses that can hardly be called intelligent: besides explanatory texts, they only provide multiple choice quizzes. This is indeed a regrettable situation in view of the successful cases including the one reported in the present paper. Through the research work so far, it may have become fairly clear what is needed in such automated tutoring systems and how to construct those systems in principle. It is, however, far less understood how we can make such systems widely available in forms that are adapted to the needs of individual courses.

The most important function of programming tutors, in our view, is to examine program codes written by students and immediately give appropriate feedbacks. This function is typically achieved either by

1. matching the submitted program with model programs prepared by human teachers, possibly through some transformations (Xu and Chee, 2003; Hong, 2004), or
2. running the submitted program against a few test cases and matching the output with the correct output (Arnow and Barshay, 1999; Odekirk-Hash and Zachary, 2001).

The latter approach is also common in so-called online judges for programming contest (see (The Association for Computing Machinery, 1977; Universidad de Valladolid, 2002) for example). Carefully selected test cases are usually sufficient for deciding if the submitted program is correct, which is the primary purpose of on-line judges. However, for instructional purposes, there are concerns besides correctness such as styles and methods. Although the first approach may be more appropriate to address these concerns, it is more difficult to determine correctness in that approach.

In MAX/C we mainly use the latter approach but enhance it by the use of a specially developed interpreter, which enables the collection of more useful information than by compiler-based executions, such as potential errors that do not surface as incorrect outputs for test cases. In addition, we use pattern-driven static code analysis of the source code. This differs from matching with model programs: the exercise authors describe what is desired and not desired of the program using a pattern language.

2 PROGRAM TESTING AND CODE ANALYSIS

As can be seen from the overview in the previous section, testing and analysis of submitted programs are major functions of MAX/C. In this section, we describe these functions in some detail.

2.1 Flow of Submission Processing

Each program submitted by a student is processed in the steps listed below. Students are required to compile and debug their programs prior to submission, but MAX/C does not assume that this is always done and is prepared for syntactic errors. Feedbacks take two forms: error messages and warnings. When error messages are generated, the processing terminates at that point and the feedbacks are given to the students. In case of warnings, the processing continues until the submission passes the test or some error messages are generated. Students have to resubmit a corrected program when they receive error messages.

1. The submitted program is parsed using the source code API. If this is unsuccessful, then error messages are generated.
2. The parsed program is analyzed statically based on patterns provided by the author of the exercise problem. If any deviations from the problem requirement are detected then error messages or warnings are generated, depending on how serious those deviations are. (This decision is made by the problem author and indicated together with the pattern description.)
3. The submitted program is executed against test cases provided by the problem author. Execution is done either by the interpreter developed in this project or gcc, a Compiler in public domain. If errors are detected, appropriate error messages are generated.

2.2 Static Code Analysis

In a static code analysis, the system detects some deviations of the submitted program from the requirements or the purposes of the problem.

For example, the subject of an exercise problem may be the use of *for* statements. In this case, the description of the task clearly indicates that *for* statements must be used in the code and the submissions are checked against this requirement. Similarly, if the purpose of an exercise is to learn the use of recursive calls, the lack of recursion in a submitted program is detected.

This analysis is driven by source code patterns provided by the author of the exercise problems. Recall the MAX approach that does not depend on general principles or artificial intelligence but rather on the mass of accumulated efforts each of which is small. To describe the patterns and associated actions, a pattern language (Takeyama, 2008) designed for this purpose is used.

2.3 Program Execution and Testing

Submitted programs are executed and tested on the server. In this phase, a run-time error or violations of the input/output specifications may be detected.

It is the responsibility of the author of the exercise problem to write a code (in the form of a Java class) to examine the outputs of the submitted program on test inputs and to determine if they satisfy the specification. This is again along the MAX approach of not hesitating to put efforts on individual contents.

We mainly use a custom-made C interpreter to execute submitted programs. The advantages of using this interpreter, over the use of a compiler, is as follows.

1. Some potential run-time errors that may not surface in a compile-and-run execution, such as an accesses to an uninitialized variable or to an unallocated memory location, can always be detected.
2. When run-time errors do occur, it is easier to collect useful information to be included in the feedbacks to the submitter.
3. It is more secure against attacks to the server embedded in the submitted program. The interpreter does not implement system calls that can be used to thwart the system.

In addition, although not utilized in the current version of MAX/C, the interpreter can be used to provide more sophisticated interactions with the submitter of the program. For example, the system could ask questions on the execution process of the submitted program, in order to confirm the understanding of the submitter, even if the submission is correct. Our interpreter is designed to facilitate such uses. Exploiting the full potential of the interpreter is part of our future research.

As an auxiliary means, a compiler is used. This is limited to the case where the execution time of the program is expected to be huge.

3 INTEGRATION INTO SAKAI

3.1 Background of the Integration

In the version of MAX/C before integration into Sakai, the following shortcomings were recognized.

1. The course contents and the system were rather tightly coupled. For example, adding new contents required the reconfiguration and redeployment of the system and there was no support for adding contents online. It was also necessary for the contents authors to have some detailed knowledge of the system.
2. There was little support for managing the drill results and the exercise problem submissions. Logs and submissions were stored in a file system and the teachers relied on offline codes that process the information stored in the file system, when they examine students' progress or read submissions. Again, there was no online support for those tasks.
3. There was no support for communication between the students and the teaching staff, among the students, or among the teaching staff members.

The first two issues are clearly serious. We also felt strongly the need of addressing the third, because of the following situation in class. The students enrolled in the course are grouped into two classes, each consisting of 50 to 70 students. Each class is taught by a teaching staff consisting of one faculty member and 3 teaching assistants. The role of the teaching staff is primarily to answer questions, as there is a lecture-type course taught in parallel, where the students learn the concepts necessary to answer the questions and solve the problems given in our course. The teaching staff is generally busy and, at peak times, cannot serve all requests of the students. To ease their burden and create more time to be spent for students whose needs are serious, there are several measures to be taken.

1. Improve the quality of the feedbacks provided by the MAX/C contents. This will help the students to resolve some of the questions for themselves, which they may currently ask the teaching staff.
2. Provide communication tools such as Wiki and FAQ, with which students may be able to resolve some of the questions. A public forum for students helping each other would also help promoting healthy cooperation among students rather than outright cheating.
3. Provide tools for the teaching staff to easily grasp progress of each student and the class as a whole. This would help the staff to appropriately allocate their limited resource.

Through the integration of MAX/C into Sakai we planned to take the second and the third measures, as well as to address the first and the second issues listed earlier.

3.2 Adding New Functions

Many of the functions to be added, described in the previous subsection, are standard in typical course management systems. Therefore, it is reasonable to employ a course management system rather than to newly develop those functions and add them to MAX/C. The course management system we have chosen is Sakai(Sakai Foundation, 2005).

Sakai is open source e-learning system that is used by more than 150 educational institutions all over the world. Each function of Sakai is embodied by what is called a *tool* in Sakai. Sakai has many tools for e-learning which we can readily use. Our strategy of integration is to port MAX/C as a Sakai tool, as well as to develop a new independent tool for course management which satisfies our special needs that are not satisfied by the general tools present in Sakai.

Most new features of the integrated system are implemented in the independent management tool described in the next section.

The main purpose of the newly developed management tool is to help the teaching staff to grasp the progresses of the students in the classroom. Two visualization features are used for this purpose. The *classroom map* is a map of the classroom where the seats of the students are displayed with colors depending on their progress. See Figure 2. Students at blue seats are the most advanced, light blue, green and yellow following in this order. Although not shown in this figure, students at red seats have made almost no

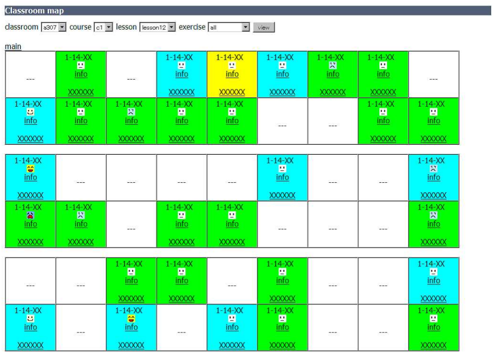


Figure 2: Classroom map.

progress and would need special assistance if they remain in this state after a certain amount of time has passed. These colors are consistent with the background colors of MAX/C pages for individual students mentioned earlier. On the other hand, the *pie chart* displays the progress of the entire class, using the same color system. The teaching staff can also share, through the management system, comments on individual students that are helpful in adapting the guidance.

4 EVALUATION

MAX/C has been used in class for more than 4 years, by more than 100 students each year. There is no doubt that the system has succeeded in immersing the students with the mass of exercises we hope them to be. To confirm the usefulness of the algorithmic feedbacks, we have conducted a survey over the students.

In this survey, we have asked students whether the drill questions and the exercise problems are helpful for learning. We also have asked whether the system is useful as a whole. 123 students have responded to this survey.

Figure 3 shows the result of the survey on the usefulness of the three types of contents: yes/no or multiple choice drill questions, trace questions, and exercise problems. To let the student focus on the merit of the feedbacks for exercise problems, the third question asks the students to evaluate the usefulness in comparison to an imaginary situation where no immediate feedbacks are given. Most of the students consider all of the three types of contents useful. They are particularly positive about the exercise problems, for which about a half of the students answered “very useful”.

However, a non-negligible number of students are not satisfied with the quality of the feedbacks given to the exercise problem submissions, as shown in Figure 4. This shows that there is a large room of improvement here.

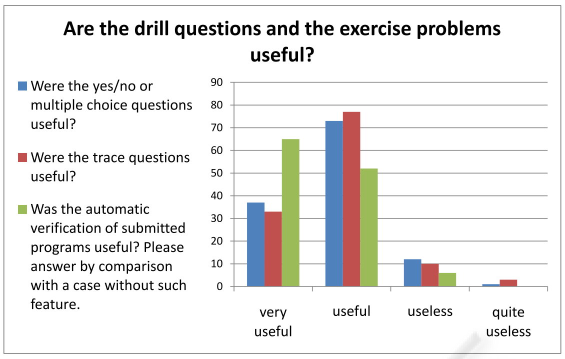


Figure 3: The survey on the usefulness of the drill questions and the exercise problems.

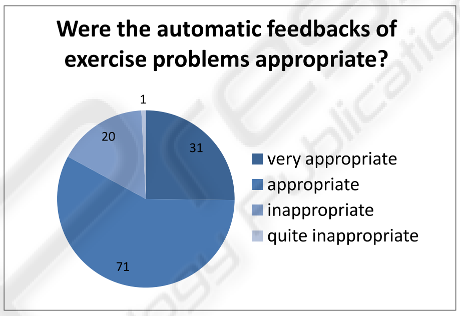


Figure 4: The appropriateness of the automatic feedbacks.

Figure 5 shows the result of the survey question asking which factors in the course are useful, allowing multiple answers. The features provided by MAX/C slightly outnumber the human support. This is natural since some students are able to proceed for themselves and are seldom in need of a help from the teaching staff. Although this is an evidence of the effectiveness of the system, we must not forget that the human support factor is still important and the system must help this support to work.

To the question on the whole system’s effectiveness (Figure 6), most students responded positively. The integration of MAX/C into Sakai has successfully been completed and the new system is currently in use. With the new management tools, it has become easier for the teaching staff to grasp the progress of the students in class. The newly added support for communication has made the interactions between students and the teaching staff easier. However, the use of newly introduced communication tools are not so widespread yet, due partly to the short time since introduction and partly to the lack of publicising efforts. It will take more time for the users to become familiar with these new features and for their benefits to be fairly evaluated.

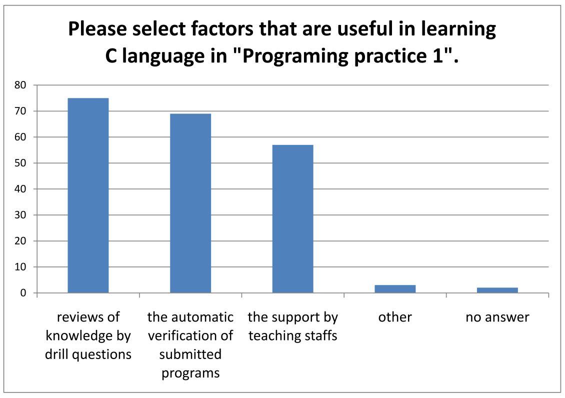


Figure 5: Useful factors for learning.

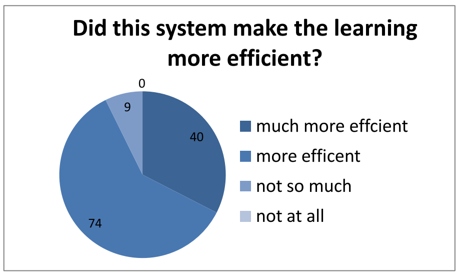


Figure 6: The overall effectiveness of the system.

5 WORK IN PROGRESS AND FUTURE WORK

The first issue raised Subsection 3.1, the lack of support for content authors, isn't resolved yet, although the integration into Sakai is a first step. This issue is being addressed in an ongoing work, where we isolate the contents of MAX/C from the system and make it into an independent repository of exercise problems, called WASABI. We have implemented WASABI as a service in Sakai and the new version of MAX/C that works with WASABI is starting to operate. We plan to develop tools for authoring on top of this repository.

We have also started a more ambitious project called MILES (Model-based Interactive LEarning Support), where not only the feedbacks to submissions but also the choice of the next exercise problem to be presented to the student is algorithmic. To do so, a proper model of each student must be built and updated during the learning process. The idea is old but there seems to be no working system along this approach available for programming education. Our approach again is to rely on the accumulation of efforts put into small parts, rather than a small number of general principles. A prototype system is being built by other members of our group and we hope to report the outcome in the near future.

ACKNOWLEDGEMENTS

We thank all of those who have contributed to this project in one way or another, especially Takateru Kojima, Tomokazu Hayakawa, Fuminobu Takeyama, and Yuta Takasugi.

REFERENCES

Anderson, J. R. and Skwarecki, E. (1986). The automated tutoring of introductory computer programming. *Communications of the ACM*, 29(9):842–849.

Arnow, D. and Barshay, O. (1999). WebToTeach: An Interactive Focused Programming Exercise System. In *Proceedings of 29th ASEE/IEEE Frontiers in Education Conference*, pages 39–44. Session 12a9.

Hong, J. (2004). Guided programming and automated error analysis in an intelligent Prolog tutor. *International Journal of Human-Computer Studies*, 61(4):505–534.

Khudobakhshov, V. (2009). Domain-specific tools for computer science. 10th Sakai Conference.

Lehtonen, T. (2005). Javala - Addictive E-Learning of the Java Programming Language. In *Proceedings of Kolin Kolistelut/Koli Calling-Fifth Annual Baltic Conference on Computer Science Education*, pages 41–48.

Odekirk-Hash, E. and Zachary, J. L. (2001). Automated feedback on programs means students need less help from teachers. In *Proccedings of the 32nd Technical Symposium on Computer Science Education*, pages 55–59.

Pillay, N. (2003). Developing intelligent programming tutors for novice programmers. *ACM SIGCSE Bulletin*, 35(2):78–82.

Sakai Foundation (2005). Sakai project. http://sakaiproject.org/.

Sykes, E. R. and Franek, F. (2004). A prototype for an intelligent tutoring system for students learning to program in Java. *Advanced Technology for Learning*, 1(1).

Takeyama, F. (2008). Static code analysis for MAX/C. Graduation project, Department of Computer Science, Meiji University.

The Association for Computing Machinery (1977). The ACM International Collegiate Programming Contest. http://cm.baylor.edu/.

Truong, N., Bancroft, P., and Roe, P. (2002). ELP - A Web Environment for Learning to Program. In *the 19th Annual Conference of ASCILITE*, pages 661–670.

Turing's Craft (2002). CodeLab. http://turingscraft.com/exers.php.

Universidad de Valladolid (2002). the UVa Online Judge. http://uva.onlinejudge.org/.

Xu, S. and Chee, Y. S. (2003). Transformation-based diagnosis of student programs for programming tutoring systems. *IEEE Transansactions on Software Engineering*, 29(4):360–384.