# Support System for Computer Programming Instruction in Group Education Environment

Masanori NAKAKUNI*, Hiroshi DOZONO+

*Information Technology Center
Fukuoka University
8-19-1 Nanakuma, Jonan-ku, Fukuoka 814-0180
JAPAN

+Faculty of Science and Engineering
Saga University
1-Honjo Saga 840-8502
JAPAN

nak@fukuoka-u.ac.jp, hiro@dna.ec.saga-u.ac.jp

*Abstract:* - The purpose of this system is to facilitate smooth instruction of computer education in group education settings used when a teacher and students gather in a single classroom. For example, in a situation where a single teacher is teaching programming to 50 students, there are cases where learning progress will vary greatly between students. Students with a sufficient understanding of class content will be able to complete an accurate program on the spot if presented with a problem during class. However, students who do not understand the course material will not be able to complete an accurate program, will have to make repeated revisions to the program, and will require significant time before completion of the program. We have evaluated the development of and created a prototype that will search for students whose progress is delayed and notify the instructor. This paper discusses the results of the experimental use of this prototype system and the future concept for this system.

*Key-Words:* - C language programming, group education, computer education, support system

## 1 Introduction

In a group education environment where a teacher and students gather in a single classroom, a single instructor or small number of instructors will provide instruction to a large number of students. In a lecture course, which involves the one-way presentation of information by the instructor, it is not problematic even when there are 100 students or more. However, in a practical application course where the instructor must provide direct instruction to each student, the one-way lecture style of teaching is not suitable. In other words, in practical application courses the instructor must progress with the class while maintaining a constant awareness of student progress but with such a large number of students this can be difficult to achieve. However, there is potential for this problem to be resolved through training using computers. For example, in a computer programming course, a common flow involves presenting a problem to students during class and having the students create a program. Students with a sufficient understanding of class content will be able to complete an accurate program on the spot if presented with a problem during class. However, students who do not understand the course material will not be able to complete an accurate program, will have to make repeated revisions to the program, and will require significant time before completion of the program. Focusing on this point, we conceived the possibility for the automated discovery of students whose progress has fallen behind the rest of the class and identify that student to the instructor to enable the instructor to provide immediate support to that student, which would lead to a reduction in the number of students unable to keep up with the class.

## 2 Current problem

Identifying students who have fallen behind in the learning process can perhaps be achieved by regularly asking students questions and observing their responses. For example, asking students who have not completed their program to raise their hands may enable the instructor to ascertain student progress once. However, this method presents a major problem. The actual process of asking students to raise hands in order to ascertain progress could cause delays among students. Frequently querying students to confirm progress could result in not only a delay in progress, but also cause some students to not raise their hand even though they have fallen behind, which would in turn make it difficult to accurately ascertain learning progress among students. For example, there could be students so intensely concentrating on program creation that they do not notice the instructor’s query. Also, among Japanese students, although in the minority, there still are introverts who are unable to express themselves. Such students might be embarrassed by falling behind the learning pace and would not raise a hand even if queried by the instructor. Conversely, among students who have completed their program, continuously responding to such queries would become annoying and many might not respond to queries. Due in part to these characteristics seen in Japanese students, ascertaining student learning progress based on a show of hands is difficult.

If querying students and confirming their response to ascertain learning progress is difficult, then it might be possible to confirm the status of learning for each student by looking at the PC screens of each student. The educational PCs we use employ a system called “Wingnet,” which allows work being done on individual student screens to be confirmed from the PC at the instructor’s desk [1]. This means that the instructor can confirm the status of student learning from the instructor desk PC. Appendix Figure A is the student PC management screen in the Wingnet system. However, there is a major problem related to the method of using this system to confirm the status of student learning. The process of confirming progress on the monitor of each student requires a certain amount of time. For example, in a class with 50 students, we can assume it would take approximately five seconds to confirm the monitor of each student. The process of confirming the monitor of each student in the class would require 250 seconds, which is more than four minutes. In a class that lasts 90 minutes, confirming the monitor of each student alone would consume 4% of total class time. Conducting this confirmation process another time would mean consuming 8% of class time. In other words, this process alone would consume nearly 10% of class time, which would reduce time for other class activities and overall make it difficult to progress the class. Furthermore, this time of five seconds is based on the assumption that the programs created by student all function properly. However, any programs with bugs will require 1-2 minutes per student to provide the instruction necessary to correct these bugs. There is sure to be students who will require bug correction for their programs so more time would be spent on individual instruction.

From these observations, we proposed a teaching support system that automatically gathers data on the student PC operations and conducts an automated and real-time analysis of this data to inform the instructor of students who may be falling behind to enable the instructor to provide immediate guidance to such students.

## 3 Prototype and software development

We developed a simple prototype software for the purpose of conducting preliminary testing prior to full-scale student support tests. This software is referred to as Teaching Support System (TSS). TSS is software that reads the status of operations conducted by the student on a PC. TSS is simple software that has the ability read, gather, and save data on mouse clicks, key input, and the frequency with which a specific key was operated during a specific point in time. Furthermore, we developed an analysis software (TSS-SOM) using Self-Organizing MAP (SOM) that is able to analysis the collected data [2, 3, 4, 5, 6, 7].

## 4 TSS log collection and analysis results

The primary purpose of these tests is to collect the key input information of each student participating in the practical course and confirm whether or not it is possible to identify indexes that make it possible to ascertain student progress. The test was conducted using the following conditions.

- Each practical course consists of approximately 50 students.
- Test involves collection of data from two classes (100 students).
- The test subjects are in beginning C programming language classes (primarily 2nd year university students).
- Computers used by students PCs running Windows 7. The students use X terminal software, which runs on Windows, to login to Linux and practice C programming language in the Linux environment.
- TSS is software that runs on Windows but all key strokes and commands to Linux are conducted through Windows. As such, key input can be monitored at the hardware level and Linux operation logs can be acquired using TSS.
- TSS runs automatically on the PC of each student from class start to end and reads all key input and

mouse click information. As such, there is no need for the student to pay attention to TSS. The system also automatically records any key input that is unrelated to coursework. Furthermore, testing and log analysis was conducted via the following procedures.

1) The student logs into the Windows PC.
2) X-terminal is launched in Windows and the student logs into Linux.
3) TSS executed.
4) Teacher and students conduct 90-minute class.
5) Following the conclusion of the class, TSS is used to collect logs.
6) TSS-SOM used to analyze logs.

The C language program used by the students for reference during the class is shown in Figure 1. Using this program as a reference, the students create their own programs based on requirements indicated by the teacher and then try to execute the program. Student key input information is gathered once per minute and saved to a log file. For a 90 minute class, approximately 90 entries are recorded to the log file. The log for each entry is equivalent to records for one minute. This single entry indicates the number of times input was conducted for each key (Fig. 2). The log in Figure 2 shows the log for a single entry. Each number is separated by a comma and the numeral to the far left shows the time at which key input information was collected. To the right is the time at which the left mouse button was clicked and to the right of that is the time at which the right mouse button was clicked. The numbers with the right of that are the times at which input was conducted for each key. This allows one to know which keys were input more or less during that time span.

An analysis of logs using TSS-SOM showed that we could ascertain the status of progress by each student. Appendix Figure B shows maps created using SOM outlined in chronological order. The numbers on the map are numbers used to identify each student. The maps show progress immediately following the start of program creation by students as well as 1 minute, 2 minutes, 3 minutes, 4 minutes, 10 minutes, and 11 minutes later. The maps allow one to ascertain which commands are being input by the students at each point in time. Program content is arranged in order from the top and includes C programming language terminology such as include, main, void, int, and double. From the map we can see which terms are being input by each student. When "return" at the end of the program is input, this status is displayed on the map to indicate that the student has finished creating the program.

```
#include <stdio.h>

int main(void) {

   int i, sum=0;

   for(i=1; i<=10; i++) sum+=i;

   printf("1+2+3+4+5+6+7+8+9+10=%d\n", sum);

   return 0;

}
```

Fig. 1 Example of C language program

```
20131029144950,9,0,0,・・・,0,0,0,0,0
20131029144955,0,0,0,・・・,0,0,0,0,0
20131029145005,0,0,0,・・・,0,0,0,0,0
20131029145010,0,0,0,・・・,0,0,0,7,0
20131029145015,0,0,0,・・・,0,0,0,0,0
20131029145020,0,0,0,・・・,0,6,0,0,0
20131029145025,0,0,0,・・・,0,0,0,0,0
20131029145030,0,0,0,・・・,0,0,0,3,0
20131029145035,1,0,0,・・・,0,0,0,0,0
                    ・
                    ・
                    ・
```

Fig. 2 Example of the key input log

## 5 Conclusion

For this experiment, we tested to see to what extent the status of progress for each student can be confirmed by reading student key input information and using SOM to analyze that information. The TSS prototype is simple software that records student key input and later uses SOM to analyze that key input. At present, we are unable to gather key input information and show SOM analysis information in real time. As such, these functions must be linked and combined into a single complete system. This system also requires functions such as the ability to identify students in need of instruction in real time and in a way that is easy for the teacher to understand.

Regarding a future concept for TSS, the primary goal is to build a system that achieves the types of functions noted above. However, this level of functionality is thought to be insufficient for the achievement of actual implementation. In addition to discovering students whose progress is slow, the ideal goal of TSS would be for the system to identify students who do not understand the course content and notify the instructor of the need for guidance for said students.

We are considering the inclusion of a function for using TSS to ascertain whether or not the student was able to complete the program on his/her own. In reality, not all students will attempt to complete a program on their own. There are some students who will try to reference programs completed by others in order to complete their own program. Furthermore, there may be students who completely copy the programs of another student and pass it off as their own completed program. For example, if we can include a function in TSS that is able to confirm student program while comparing the progress of students sitting next to each other, it would be possible to ascertain whether students have completed a program based on their own ability or through the assistance of a friend sitting next to them. By discovering such students with insufficient understanding and providing instruction, it may be possible to improve learning efficacy during class. As shown in Figure 3, we hope to improve efficacy of the instruction process whereby during actual use of the system the instructor is equipped with a tablet that displays TSS analysis results in real time, thereby allowing the instructor to seek out and move to students in need of assistance.

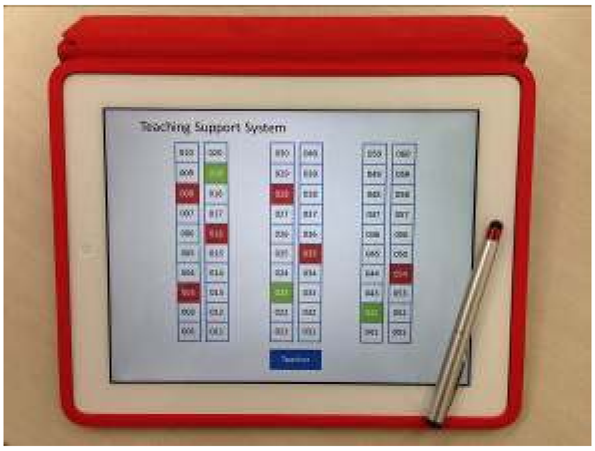

Fig. 3 Example of the system for instructor

**Acknowledgements**: This work was supported by JSPS KAKENHI Grant Number 24500279.

*References:*

[1] Computer Wing Co.,LTD, Wingnet, Support for education system, http://cw.co.jp/ .

[2] T. Kohonen, *Self Organizing Maps*, Springer, ISBN 3-540-67921-9, 2001.

[3] H. Dozono, An Algorithm of SOM using Simulated Annealing in the Batch Update Phase for Sequence Analysis, *Proceedings of 5th Workshop on Self Organizing Maps*, 2005, pp.171-178.

[4] L. Huang, Consumer's Purchasing Behavior Analysis Based on the Self-Organizing Feature Map Neural Network Algorithm in E-Supply Chain, *Neural Information Processing*, Lecture Notes in Computer Science Volume 4232, 2006, pp.1022-1029.

[5] H. Dozono, M. Nakakuni, An Integration Method of Multi-Modal Biometrics Using Supervised Pareto Learning Self Organizing Maps, *Proceedings of 2008 International joint Conference on Neural Networks*, IEEE Press, 2008, pp.603-607.

[6] H. Dozono, An Algorithm of SOM using Simulated Annealing in the Batch Update Phase for Sequence Analysis, *Proceedings of 5th Workshop on Self Organizing Maps*, 2005, pp.171-178.

[7] H. Dozono, M. Nakakuni, The Supporting System for Teaching C Language Based On Self Organizing Maps, *Proceedings of 14th International Symposium on Advanced Intelligent System*s, 2013, CD-ROM.

## Appendix

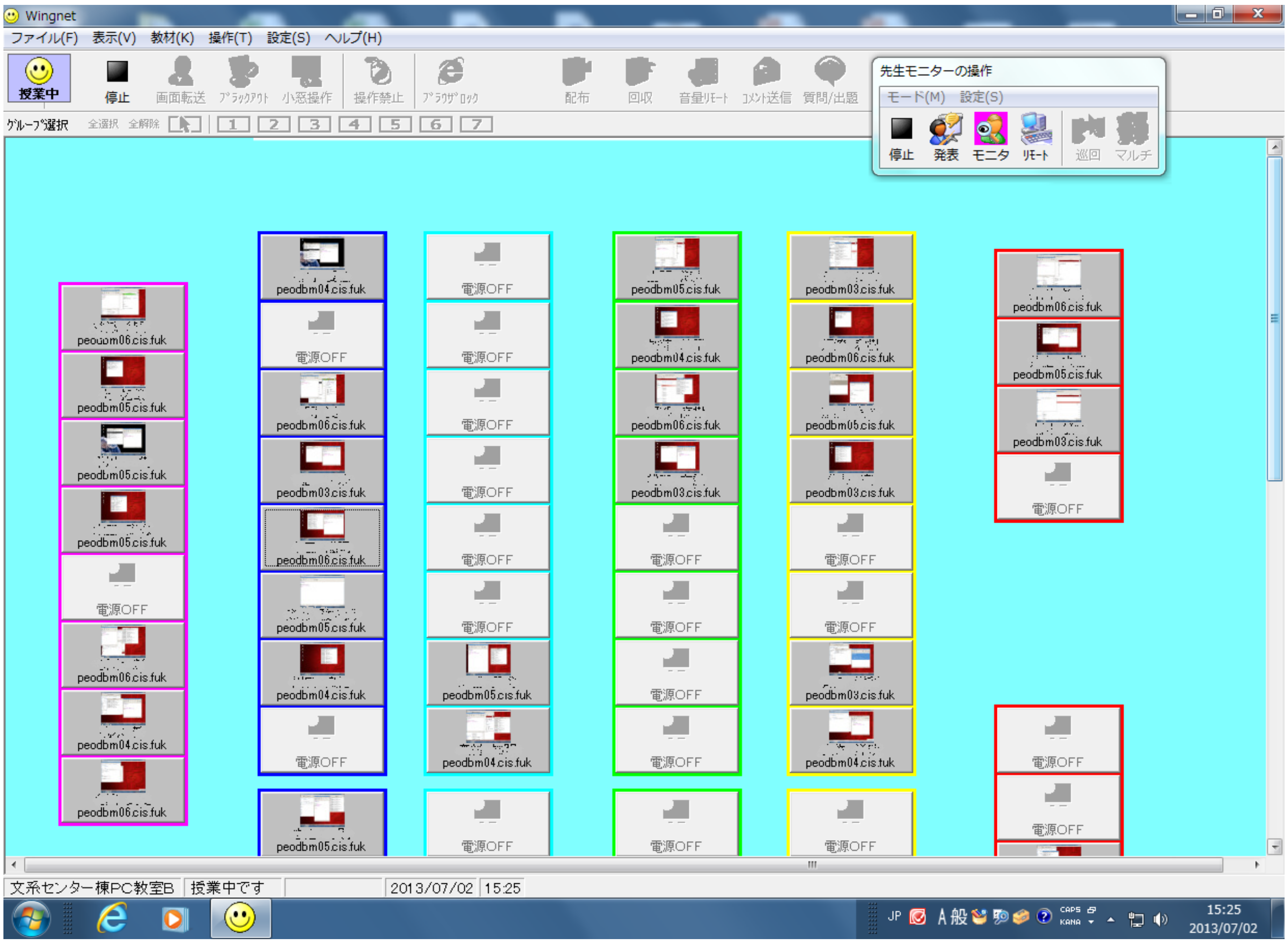

Fig. A The student PC management screen in the Wingnet system

A: main B: void C: int D: for E: printf F: return, 1〜40: Student's Number

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 18 | | | | 5 7 | | | A 13 | | C 12 |
| | | 35 | | | | | | | |
| 21 | | | | | | C 19 | | | 23 |
| | | | 33 | | | | | | |
| 10 | | | | | | 1 | | | 9 27 |
| | | | | 30 | | | | | |
| 29 | | 22 | | | 31 | | | C 16 | C 8 |
| | | | | | | | C 7 | | |
| | | | 26 | | | | | | |
| 36 | | | | | C 15 | C 25 | | C 32 | C 24 28 |

**Map of the students after 1 minutes**
The student who inputs "main" and "int" can be found on this map and students show the state of having begun to write a C program.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 5 | 36 | | D 26 | D 28 | D 19 | D 24 | | D 7 15 |
| | | | | | | | | | D 8 |
| 31 | | | | | D 23 | | | | D 8 |
| | | | 3 | | | | D 16 | C 27 35 | |
| 11 | | | | | | | C 25 | | |
| | | | | | C 22 | | | | C 33 |
| 39 | | | 37 | | | | | | C 29 |
| | | | | | | | C 18 | | |
| | | | | | | | | | C 17 |
| 4 | 20 | | C 12 | | C 2 | C 34 | | C 10 21 | C 1 13 |

**Map of the students after 2 minutes**
The student who inputs "int" and "for" can be found on this map.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 38 | 3 | 15 | 26 | | D 7 25 | D 35 | D 8 | D 27 | D 21 |
| | | | | | | | | | D 1 |
| | | | | | D 9 | | | | D 19 |
| D 36 | D 17 | | 23 | | | | | D 2 | |
| | | | | | | | | | |
| 37 | | | | | D 34 | | C 29 | | C 18 33 |
| | | | 22 | | | | | | |
| 24 | | | | | | | | C 30 | |
| | | | | | | | | | |
| C 28 | C 10 | | C 32 | | | C 20 | | | C 11 31 |

**Map of the students after 3 minutes**
The student who inputs "int" and "for" can be found on this map.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 38 | 18 13 | 15 | 9 | | D 11 | | 2 | D 21 | D 1 20 |
| 30 | | | | | | | | | |
| | | | 27 | | | | | | D 28 |
| | | 23 | | | D 31 | | D 8 | | D 7 |
| D 29 | | | | | | | | | |
| | | | | | | | | | D 35 |
| 36 | | 17 | | | D 22 | | | | |
| | | | | | | | | | D 25 |
| 10 | | | | | | | | | |
| | | D 12 | | | D 32 | | | | 19 |

**Map of the students after 4 minutes**
The student who inputs "for" can be found on this map.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 19 | | 13 | | 39 | | F 15 | | | F 17 |
| 25 | | | | | | | | | |
| | | | | F 37 | | | | | F 29 |
| | 19 | 38 | | | | | | | |
| | | | 31 | | | | 14 | | 33 |
| 18 30 | | | | | | | | | |
| 2 | | | 26 | | | | | | |
| 24 | | | | | | 36 | | | F 10 |
| 7 11 | | | | 12 | | | | | |
| 35 | 23 | | F 32 | | | F 20 | F 28 | | F 27 |

**Map of the students after 10 minutes**
The student who inputs a "return" can be found on this map and students show the state of having finished writing a C program.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 19 | | 8 | 15 | | 2 37 | | | | E 38 |
| 18 34 | | | | | | | | | |
| | | | | | | | | E 14 | |
| | 12 | 36 | 7 | | E 4 | | | | |
| 16 | | | | | | | D 10 | | E 39 |
| 9 25 | | | | | | | | | |
| | | 26 | E 30 | | 27 | | E 13 20 | | |
| 23 31 | | | | | | | | | |
| 11 | | | | | | | | | E 28 |
| E 17 33 | 35 | 1 | 5 | | 24 | | 32 | | E 29 |

**Map of the students after 11 minutes**
The student who inputs "int" and "printf" can be found on this map. And we can find out the student who cannot complete writing a C program.

Fig. B Mapping of students' progress