Building Confidence and Skills: A Prep Course for Computer Programming

Linda Head, Department of Electrical and Computer Engineering, Jennifer Kay, Department of Computer Science, John Schmalzel, Department of Electrical and Computer Engineering, Glenn Arr, Christopher Foster, Steven McDermott, Michael Sterner, Kenneth Whelan, and Jason Wollenberg, ECE Student Teaching Team, Rowan University, 201 Mullica Hill Rd., Glassboro, NJ 08028

Abstract

Students entering the Rowan University College of Engineering arrive with a very diverse set of computing skills. Typically, they are familiar with the common packages such as Microsoft's Office suite and most have used e-mail and played computer games of some type. However, a significant portion (greater than 50%) have not had rigorous programming experience. Since our common first year program has a C++ programming course in the second semester, we are concerned about both our students' preparation for this course and their level of confidence in mastering the basics of computer programming.

To meet the needs of our students we have initiated a Programming Preparation Course. This course is a collaborative effort of faculty in the College of Engineering and the Computer Science Department in the College of Liberal Arts and Sciences and a dedicated teaching team of six junior level Electrical and Computer Engineering students. Together we have designed a curriculum that will be taught outside of class time and will introduce the students to some fundamental concepts of computer programming. There are three sessions offered to the first year students, each is one hour long and focused on a limited topic set. The topics which we are using to introduce the fundamentals of programming are: (a) variables, output and the "if" statement; (b) loops (while and for) and input; and (c) function calls. The curriculum is based on a "show and do" method. The teaching team prepared a set of simple programs that the first year students ran and then modified in order to gain some skill and insight into how the programming sequence works.

This year is the first time that we have attempted this type of class. We offered the course over a four week span during the second half of the Fall, 2000 semester. Thirty-seven students enrolled in the course. We will be tracking the progress of these students and a control group who also have not had prior programming experience to assess the effectiveness of our initial course offering. In this paper we describe the program and report the current progress of our assessment.

Introduction

Introducing students to engineering and orienting them to engineering practice is the usual goal of first year programs in an engineering school. In addition, this introduction to the disciplines

that is offered at the school provides the students who have not yet decided on their major an opportunity to make an informed choice about the direction of their education. At Rowan University's College of Engineering we try to provide content that will aid our students in their future engineering courses along with this typical disciplinary introduction. To this end, the first semester of Freshman Clinic (our version of Intro. to Engineering) consists of three-week long laboratory-based modules in each of our four disciplines. We are committed to including significant content in these modules as well as demonstrating to the students the extent to which all engineering disciplines overlap. The multidisciplinary aspects of our program are crucial to the type of education that we provide and are one of the reasons that we maintain a common curriculum for all of our students during their first year. At the end of their first year we hope to have given them not only a sound background in the fundamentals of mathematics and science, but also a solid appreciation of the skills that are required for engineering practice.

One of the most fundamenta skills required in today's high-technology workplace is the ability to use a computer. We all are aware of the extent to which computers have permeated our everyday lives. So it is, of course, important that our student know how to use software packages to perform common tasks. However, this is not adequate for engineering problem solving. Engineers must have a more extensive understanding of the fundamentals of computing functions so that they can take full advantage of the powerful computing resources available to them, regardless of their discipline.

During the second semester of the freshman year our students are required to take a computer programming course. This course emphasizes programming methodology, algorithms and simple data structures. At present the C++ programming language is used to introduce these concepts. This course is a standard part of the Computer Science curriculum and the Engineering students take it along with Computer Science majors. Prior programming experience in any programming language is expected for this course. However, because of credit hour restrictions in a tight engineering curriculum, we are not able to offer our students an introductory course in programming in the first semester of their freshman year. This means that those students who do not enter engineering with some programming experience (approximately 60% of our students in the class that entered Fall 2000) are at a disadvantage. We have tried a couple of "fixes" for this problem. The first, giving an introduction to C++ during the Electrical and Computer Engineering module of Freshman Clinic, did not work due to the limited amount of time available for the task and the wide range of preparation that the students bring to the sessions. The second, using MATLAB© to introduce simple programming concepts, also suffered from the limited time available and differences in student comprehension. The students who had some programming experience found the exercises trivial, and those with no programming background did not see the point of using MATLAB© for a problem that a spreadsheet could have solved more efficiently.

We decided that a solution might be more effective if we targeted only the population of students who needed fundamental instruction in programming. This past fall we put together a voluntary program for those freshmen who needed instruction in programming basics. The program was designed, implemented and delivered by six junior level students in the Electrical and Computer Engineering program. The students used the program to obtain credit for their own Engineering Clinic activities. They were asked to think about the best ways to introduce programming

principles to freshmen who had not had any programming experience. These freshmen were typically not the students who were heading for an Electrical and Computer Engineering major and had some anxiety over the prospect of having to learn how to program. Looking back to their own experiences and those of their "non-EE" friends, the instruction team concluded that the best way to learn how to program is to "just do it." They developed a totally hands-on Programming Preparation Course.

The Programming Preparation Course

The curriculum for this program was guided by the "pre-test" that is given by the Computer Science Department to all students who are beginning their programming course. This test addresses the topics that the students are expected to be familiar with at the beginning of the course. These topics are:

1. Typed variables and assignment statements
2. Input and output commands
3. Program structure
4. Conditional statements (e.g., "if")
5. Loops (e.g., while)
6. Function calls

The six-member teaching team met with faculty in the Computer Science Department to learn the best techniques for introducing these concepts. The Programming Preparation Course was divided into three sessions:

Session	Topics
One	Variables, Program structure, The "if" statement, Input and output
Two	Loops, for and while
Three	Function Calls

Each sessions consisted of a short lecture followed by extensive hands-on activities. To illustrate the technique, Session One will be described here in detail.

The instructors began by teaching simple output techniques. Immediately the students were allowed to experiment with a variable in an actual program. In order to avoid the complexities of the Microsoft Visual C++ environment, the students only worked within the context of a program that was already loaded and ready to run. The first program that they saw is shown below as Figure 1. Everything necessary to compile and run the program was already set up, our goal was not to teach them C++ but to show them how the simple components, such as text output, are handled in a computer program. Note that the students were instructed to only modify the program in the body so that they did not have to worry about any of the details associated with the "overhead" structure of C++. They only manipulated statements related to the topics addressed in the session. The first task they were given was: Add to the program so that it also says "Welcome to Introduction to Programming." Then they were encouraged to play

with this short program to make themselves comfortable with text output to the screen and the linefeed command.

```
//Day1 Sample 1
//
//

//preprocessor statements
#include "iostream.h"
#include "stdlib.h"

int main()
{

/**************************************************
**************************************************

DO NOT MODIFY ABOVE THIS SECTION

**************************************************
**************************************************
*********  DO NOT MODIFY ABOVE THIS LINE    ********/

        //output to the screen
        cout<<"Hello World";
        cout<<endl;



/*********  DO NOT MODIFY BELOW THIS LINE    ********
**************************************************

DO NOT MODIFY BELOW THIS SECTION

**************************************************
**************************************************

**************************************************/
        return 0;

}//end main
```

Figure 1 First sample program.

Next, the instructors defined the concept of a variable as an information storage box that requires labeling with a type (e.g., “int”) and starts with an unknown quantity inhabiting the box. The students then moved on to working with the programs shown in Figure 2. In each of these programs the students addressed a new topic; in Program 2, variable declaration, assignment and computation using variables; in Program 3 the “if” statement and relational operators were introduced. In each program the previously used components were repeated so that by the time the students understood and used Program 3 they were combining text output (Program 1) with numeric variables, computation, and relational operators (Program 2) with the “if” statement. By using this modular technique where the students only had to change portions of the body of the program, they were able to concentrate on learning the programming concepts. During Sessions Two and Three the same method was used to introduce the students to Loops and Function Calls. By the end of the hour on the third session the students were writing programs to solve problems.

Much of the success of the sessions could be attributed to the individual attention that the students received from the six Junior instructors. Although approximately 37 students attended

the sessions (30% of Freshman class), each session was delivered three times on three different days so that as many students as possible could fit the Programming Sessions into their schedule. This made it possible for the six instructors to provide individual attention to the students.

```
//Day 1 Sample 2
//
//

//Preprocessor Statements
#include "iostream.h"
#include "stdlib.h"

int main()
{

/**************************************************
*
**************************************************
*

DO NOT MODIFY ABOVE THIS SECTION

**************************************************
*
**************************************************
*
*********  DO NOT MODIFY ABOVE THIS LINE
********/

        // Declare variables
        int a;
        int b;
        int c;

        //Initialize Variables
        a=2;
        b=5;
        c=a+b;

        //Output to the Screen
        cout<<"a + b is ";
        cout<<c;
        cout<<endl;


/*********  DO NOT MODIFY BELOW THIS LINE
********
**************************************************
```

```
//Day 1 Sample 3
//
//

//Preprocessor Statements
#include "iostream.h"
#include "stdlib.h"

int main()
{

/**************************************************
***
**************************************************
***

DO NOT MODIFY ABOVE THIS SECTION

**************************************************
***
**************************************************
***
*********  DO NOT MODIFY ABOVE THIS LINE
********/

        //Declare Variable
        int age;

        //Initialize Variable
        age=21;
        //Conditional Statement
        if( age >= 18)
        {
                //Output to the Screen
                cout<<"You may vote";
                cout<<endl;
        }
        //Output to the Screen
        cout<<"Have a nice day.";
        cout<<endl;


/*********  DO NOT MODIFY BELOW THIS LINE
********
```

Figure 2 More practice codes for Session One

Assessment

At the completion of the Programming Preparation Course the students were sent a set of survey questions to evaluate the sessions. Ten of the thirty seven students returned the surveys, a much lower number than was expected, however, we are planning to contact these students again during the spring semester to follow up on their progress in the computer programming course. Table 1 shows the extreme response options and the average response to each of the survey questions.

Table 1 Survey Responses

1. When the help sessions began my knowledge of computer programming was:

1 none	5 good	Average Response = 1.8

2. The instructors provided clear explanations of the concepts.

1 disagree	5 agree completely	Average Response = 3.8

3. The handouts were understandable.

1 disagree	5 agree completely	Average Response =3.9

4. The amount of time spent on each topic was just about right.

1 disagree	5 agree completely	Average Response = 3.9

5. There was adequate individual attention available during the help sessions.

1 disagree	5 agree completely	Average Response = 5

6. The example problems worked during the help session were:

1 too easy	5 too hard	Average Response = 3.1

7. The times that the help sessions were offered were convenient.

1 disagree	5 agree completely	Average Response = 4

8. I would like to have students available for programming assistance during the CS&P course next semester.

1 disagree	5 agree completely	Average Response = 4.9

9. I would recommend this session to a new student next fall.

1 disagree	5 agree completely	Average Response = 4.7

10. Please take another look at the pre-test (attached).
I am confident that I could work the problems on the pre-test after attending the help sessions

1 disagree	5 agree completely	Average Response = 3.4

Most of the students who responded had very little background in computer programming. Question 1 yielded an average response of only 1.8. From the response to question 10, we can conclude that at least some of the students who did not have a good programming background gained in both skill and confidence as a result of the Programming Preparation Course. Seven of the ten respondents said that the best thing about the class was the individual attention that they received from the instructors. This seems to have been the key to the initial success of the program. Individual attention from another student who understands their difficulty helped make them feel comfortable with the material. All the students said that they would be likely to recommend this program to next year's Freshmen and all hoped that students would be available for help-sessions next semester during the programming course.

Conclusion

In the final analysis it will be the students' performance in their computer programming class that will be the best assessment tool for measuring the success of the Programming Preparation Course. The first semester of college is difficult for all students. They are faced with many new

challenges and their level of preparation is not always adequate. It is a very important aspect of our introductory program to assess each student's readiness for the tasks ahead and work with them to make sure that they are as successful as possible. The Programming Preparation Course described here was completely voluntary, each of the thirty-seven students who participated did so on a weekday evening or a Saturday afternoon. The benefits of their effort and that of the instructor team that developed this course have the potential of increasing our retention in the programming course during the spring semester.

LINDA HEAD
Linda Head has taught Electrical and Computer Engineering at Rowan University since 1998. Prior to Rowan she was in the Electrical Engineering Department at SUNY-Binghamton. She has been involved in teaching Freshman Engineers since coming to Rowan and is also faculty advisor for SWE and a participant in AWE, a local summer program for middle school girls.

JENNIFER KAY
Jennifer Kay joined the Computer Science Department at Rowan in 1998. She received her Ph.D. from Carnegie Mellon University, and prior to Rowan she was a member of the Artificial Intelligence Laboratory at Lockheed Martin's Advanced Technology Laboratories. Her research interests include Robotics, Computer Science Education, and Human Computer Interaction.

JOHN SCHMALZEL
John Schmalzel has been at Rowan University since 1995, currently serving as chair of the Electrical and Computer Engineering Department. He has been active in the development of Rowan's new ECE curriculum, with particular interest in the Engineering Clinics, a multdisciplinary, 8-semester sequence. His other interests include instrumentation and laboratory development.