Reviewing the Success of a Freshman Programming Course

Dan Budny[1], Laura Lund[2], Karen Bursic[3] and Natasa Vidic[4]

Abstract - **Our experience with teaching C programming to freshman engineering students has revealed that the most difficult concepts for students to learn are arrays, language syntax, designing the organization of the program, looping and branching, function calls, and debugging. In a typical C programming course the instructor must address all of these conceptual challenges simultaneously, because of the nature of the language. To help solve this problem, we have addressed the various conceptual difficulties by using different software packages to introduce each topic independently. For example, we have discovered that EXCEL can be used to visually demonstrate the concept of an array, as well as matrix operations and data input. The concept of designing the layout of a program can be introduced effectively with HTML, while file addressing and accessing of remote data can be studied with UNIX. The concepts of looping and branching can then be introduced with MATLAB. Once the students are familiar with the use of EXCEL, UNIX, HTML and MATLAB, the introduction of C is less overwhelming for the students to understand. This paper describes how we introduced this new teaching concept into the University of Pittsburgh Freshman Engineering computing courses, and provides the results of a survey designed to determine current freshman student perceptions of the effectiveness of the curriculum, and upper class student experiences in using what they learned from these courses.**

Index Terms — *Freshman, Programming skills, Problem solving.*

INTRODUCTION

A review of most tables of content of an introductory text in C will reveal the same basic course layout, with the following standard outline [1 - 3]:

1. Computer fundamentals
2. C Basics - variables, arithmetic operators, math functions, input/output
3. Decision making - branching
4. Looping
5. Functions
6. Arrays
7. Pointers

There have been a number of papers presented at both the ASEE annual conference and the Frontiers in Education conference regarding techniques to improve the quality of instruction in the area of computer programming [4 - 9]. The debate has covered many topics regarding the success and failure in the instruction of programming languages. Some researchers [4] have stated that the constraints of the academic environment rarely provide an opportunity to replicate the size and complexity of a typical industry project. Thus, instead of using the above course outline the course should follow other directions such as the Software Engineering Body of Knowledge (SWEBOK), that breaks down the topics into the following areas: Software Configuration Management, Software Construction, Software Design, Software Engineering Infrastruction, Software Engineering Management, Software Engineering Process, Software Evaluation and Maintenance, Software Quality Analysis, Software Requirements Analysis, and Software Testing. This is more of a "big picture" view of the subject.

Fincher [5] has noted that traditionally programming has been taught as the professors learned it, via syntax, through the vehicle of a single language. The limitations of this approach – that students get bogged down in the specifics of the chosen form, that they see programming as "fighting the compiler" – are frequently bemoaned and yet, as frequently, this is the approach that dominates undergraduate teaching. She believes very little attention has been paid to the rationale which explains why we teach the subject.

Other researchers [6] found it helpful to stop teaching formal languages during the first semester and instead, stress the four basic steps to problem solving (analyze, design, code, and test), with an emphasis on algorithm development. This was supported by others [7] that found a value in integrating programming and problem solving instead of teaching each separately.

[1] Dan Budny, University of Pittsburgh, Department of Civil Engineering, 323 Benedum Hall, Pittsburgh, PA 15261 budny@pitt.edu
[2] Laura Lund, University of Pittsburgh, Department of Bioengineering, 323 Benedum Hall, Pittsburgh, PA 15261 lwlund@pitt.edu
[3] Karen Bursic, University of Pittsburgh, Department of Industrial Engineering, 1044 Benedum Hall, Pittsburgh, PA 15261 kbursic@pitt.edu
[4] Natasa Vidic, University of Pittsburgh, Department of Industrial Engineering, 1032 Benedum Hall, Pittsburgh, PA 15261 nvidic@pitt.edu

However, others [8] found in order to learn through redundancy, fundamental concepts must be introduced early. In many textbooks, topics such as development of user-defined subprograms, pointers, data structures, and abstract data types are not covered until later chapters. This precludes the possibility of really learning these concepts in the course.

Over the past few years the authors have been struggling with many of the same problems and concerns discussed in the literature. We have found repeatedly that students without any prior programming experience are unable to learn all of the C programming concepts and demonstrate adequate programming proficiency in one 15 week semester.

OUR SOLUTION

In the initial development stages of our freshman engineering course curriculum, we struggled with the traditional approach to teaching C. The primary problem is that all introductory C text books treat the subject similarly in terms of the order in which material is presented, as listed in the previous section. However, we have discovered that the difficulties that the students have with C appear to be centered on the concepts of:

- Arrays (indexing and constructing)
- Looping & Branching
- Program design, organization and modularization
- Functions
- Debugging

In addition, students seem to have difficulty mastering the detailed C language syntax requirements while simultaneously attempting to master the more complex concepts listed above. A review of the text books reveals that these concepts are typically taught during the latter portion of a one semester course. Thus, the problem becomes how to move this material to the beginning of the curriculum.

Our solution was to design a two semester course sequence that uses EXCEL, UNIX, HTML and MATLAB to introduce these concepts during the first semester. The first 7 weeks of the second semester is spent introducing functions in MATLAB, and on intensive programming assignments designed to help students master the above listed concepts. The C language is then taught during the final 8 weeks.

BACKGROUND

Engineering 0011- Introduction to Engineering Analysis

All students are required to take four core engineering courses during their first year. There are two zero-credit seminar courses and two three-credit introductory problem solving courses that make up this core. ENGR0011 is the required first semester three credit course for all freshmen engineers, which meets twice a week for 2 hours in a computer-equipped classroom. The goals of ENGR0011 are to:

- Teach the basic computer skills, and their role in problem solving,
- Develop programming concepts and skills using EXCEL, UNIX, HTML and MATLAB,
- Introduce teamwork,
- Improve writing and communication skills,
- Illustrate the use of software packages (MSWord, PowerPoint) in communications,
- Ensure that students understand how material in the basic sciences and mathematics is used by engineers to solve practical problems of interest to society.

In this first semester course, the students are introduced to a number of the various concepts and skills required to be successful in engineering, including not only mathematical techniques such as curve fitting and basic statistical analysis, but team work, communication, and research skills as well. These concepts and skills, and the concept of programming as a problem solving tool are demonstrated and developed through the use of several software programs beginning with EXCEL, then progressing through UNIX, HTML and MATLAB. Assignments have been structured for each of these programs to not only demonstrate how they can be used to solve engineering problems, but also to help develop an understanding of the programming concepts ultimately need to program in C. The Table below summarizes the programming concepts that are addressed through lectures

TABLE 1:
SUMMARY OF PROGRAMMING CONCEPTS INTRODUCED THROUGH EXCEL, UNIX, HTML AND MATLAB

Unix	Excel	HTML	MATLAB
Syntax	Arrays	Syntax	Arrays
File addressing	Data input	Debugging	Looping/Branching
Accessing remote data	Logic	Modularization	Syntax
	Accessing remote data	Program design & organization	Functions (& modularization)
	Modularization		Program design & organization
			Debugging

and assignments in EXCEL, UNIX, HTML and MATLAB.

A portion of this course is taught interactively in a cooperative learning environment where the students work in teams to solve the course requirements. This helps meet the goal of improving teamwork and communication skills.

The course also has a significant writing component that requires the students to research and present a topic in engineering and to discuss how it impacts them personally, as well as the engineering profession and society. The writing assignments do not introduce programming topics, but they are structured in a progression of assignments which are intended to teach the student how to outline a problem, research the problem and compose a logical solution. This valuable skill is analogous to that needed to develop and write an effective computer program.

It is the experience of the faculty that the majority of incoming freshman students know very little about the actual operation of a computer or the use of computer software as problem solving tools. The students are good at using packages such as AOL instant messenger, Facebook and finding music files on the web, but when it comes to organizing files in directories, or organizing their thoughts into a structured program, the vast majority of the students are initially lost. Thus, the main focus of ENGR0011 is to begin the process of structured thinking, and to develop a sense of confidence in learning and using new and different software programs.

Engineering 0012- Introduction to Engineering Computing

ENGR 0012 is a required second semester three-credit course for all freshmen engineers. It meets twice a week for 2 hours in a computer-equipped classroom. It has the following overall goals:

- To teach how to program a computer using a number of general-purpose programming languages.
- To promote and encourage good programming practices including a top-down approach.
- To illustrate the role of computer programming in solving engineering problems.

This course is a continuation of ENGR0011. The course material completes the coverage of MATLAB, and then moves to the use of "C" to solve a given engineering problem. ENGR0012 is the next step in the development of the students' problem solving skills. During the first semester students are introduced to the various concepts and skills required to be successful in engineering, in this course they are required to use these skills.

In both ENGR0011 and ENGR0012 open-ended homework projects related to engineering topic areas are assigned. Students have several options and must make efficient choices in order to solve the problem(s) at hand. These projects are intended to introduce engineering, and challenge students' judgment and creativity as well as their problem-solving abilities.

EXCEL

ENGR0011 begins with a four week introduction to EXCEL. The learning objectives for this component of the course are introduction to the basic data entry, and construction of equations, along with the introduction of the concepts of arrays, matrices, linear algebra, logic control and function calls. During the process of covering this material we also introduce curve fitting (linear, semi-log and log-log), basic statistics, goal seek and a number of matrix operations.

The basic concept that we try to introduce to the students is that a spreadsheet is nothing more than a matrix or a two dimensional array. Thus, we can teach the concepts of arrays using a graphical approach with EXCEL. The only disadvantage of EXCEL is that it references the cells by a {column, row} numbering scheme instead of a {row, column} scheme that is used in MATLAB and C. However, we use this difference to emphasize the concept of referencing an element within an array.

Since we started teaching EXCEL, we have observed that the significant majority of students do not understand matrix operations, or have never been exposed to matrices before. This explains why students have a difficult time understanding arrays when discussed in C. The introduction of matrices in EXCEL is unquestionably helping the students understand the concept of arrays in both MATLAB and C.

Another concept that we introduce with EXCEL is importing data from a file. EXCEL allows the user to input data from an external file, this is directly related to the "input and load" commands in MATLAB and the "fscanf" command in C. The commands are different, however, the basic concept of importing external data into a program is the same in all languages. Transferring data from a file to a variable within a program is a very difficult concept for students to understand, the use of EXCEL to introduce this concept has improved the ability of the students to perform this task in formal programming languages,

The concept of "functions" is also introduced by using multiple worksheets in EXCEL. By having the students put data on multiple sheets and then addressing the data between sheets, we introduce the concept of different "workspaces" or a very basic form of a function call. For example the data may be stored on one sheet, and calculations performed on another sheet. We clarify that we are passing the data from the data sheet to the calculation sheet. This concept of passing data between "programs" can then be further explained by plotting the data and storing the figure on a third sheet. Simple things like naming the column with different names in the different sheets is the very basic beginning to different variable names between the main and function in "C".

EXCEL also allows us to introduce the concept of order of precedence in writing an equation. This is one of the basic areas of syntax errors in any language. The color coding of the cells that EXCEL performs allows the students to visualize the equation.

Finally, EXCEL allows us to begin to introduce the concept of a logical operator. The “AND” and “OR” operators along with all of the rational operators can be explained in the filter operations within EXCEL. In addition to Filtering, EXCEL also has functions such as IF, AND, OR, COUNTIF, VLOOKUP and other functions that make a decision. Thus, you can use a spreadsheet to begin the concepts of branching in a program and the decision making process. We can also introduce the concept of a required “argument” in a function call using EXCEL commands.

In summary, EXCEL is a very simple and user friendly software that allows us to introduce: arrays, input/output, functions, logic and syntax issues in a manner that is not feasible in C. It is a small step but we have found out that this basic introduction is helping the students understand the big picture of what an array and function are and why we use them.

UNIX AND HTML

The next step after EXCEL was the introduction of MATLAB. However, we found that the students had a number of basic concept problems: The first problem was that of the concept of the "path" or file management. MATLAB uses m-files to store the main program and all the function files. To execute the program, the user must set the path to all the files. That is, the user must understand the concept of file management. Once again we discovered that this concept is not easily understood by most freshman. Another concept that was hard for students to grasp was how to layout the design of a program with branching, looping and function calls and while doing this how to debug a program.

To help with these concepts, we decided to spend four weeks teaching HTML coding before introducing MATLAB. The HTML component had a number of learning objectives. A standard web page with links requires the knowledge of file management and directories. Thus, since we understood that file management was difficult for students, we used HTML as a reason to introduce UNIX and the basic file management structure used with UNIX. It is impossible to design a web page with both relative and absolute link addresses without some knowledge of the file management.

In addition, HTML is a very simple form of programming. There is no branching or looping, so the logic is very simple, but we discovered that the layout of a web page allowed us to introduce the basic concepts of program layout. We require the students to code the web pages in HTML using an editor. We step through the use of PICO, followed with WordPad plus the use of ftp, and finally the use of the Microsoft Visual C++ program editor. By using an editor instead of any of the web design programs, the students are introduced to the practice of laying out code line by line. We advise the students that the reason we teach web page design is not only for the purpose of teaching HTML, but to teach students the concept of writing code. The use of the Visual C++ editor also allows us to introduce the concept of debugging the code. The Visual C++ editor has a color code feature for tracking the HTML tags and also automatically tabs the lines. Again, these simple steps are designed to target problem areas we have observed when teaching a traditional C course.

We stress the proper use of comment lines, white spaces and indenting code. In addition, the start and end tags in HTML can be directly related to the "{ and }" brackets in C or the use of the "end" command in MATLAB. By using HTML to explain the concept of syntax we begin the process of learning to be detailed in the writing of a code. By requiring the students to design their web pages using tables, we can reinforce the need for paying attention to the details. Tables in HTML offer a great teaching tool. It allows the student to debug their code with a graphical output instead of a series of error statements. Either the page looks like it was designed to look or it does not. If it does not they must find the errors in their HTML code.

HTML also offers the instructor the opportunity to introduce function calls, by converting a web page into a frame set. A frame set is nothing more than a function call. You have an index file that calls a number of other files. The web page output is not the result of the index file but the result of the files called by the frame set. We then introduce style guides as another form of function calls. Again, the external or internal style guides are a form of passing control of the final web page appearance between different files. The text in the web page comes from one file while the appearance properties come from a different file. This concept allows us to continue the discussion started in EXCEL, with passing control of the output from one sheet to the other in EXCEL to the transfer from one file to another in a frame set or a style guide.

Finally, the main reason we decided to introduce HTML was because the students just love it. Every freshman wants to create their own web page. We found that they will do anything if the result is a "cool" web page. Thus, we have hidden the introduction of UNIX, proper program layout, introduction to syntax, introduction to debugging and a continuation of the concept of function calls all into an assignment they love to work on. They will put 100 hours into a web page without complaining, but if you ask them to put 2 hours into a C program they will quit at the first sign of an error statement. Thus, one of the biggest values in teaching HTML is the confidence that the student gets in their ability to debug code.

MATLAB

MATLAB is a technical computation software package maintained by The Math Works, Inc. (Natick, MA). We spend the last five to six weeks of the first semester ENGR0011 and the first seven weeks of the second semester ENGR0012 courses teaching MATLAB.

Learning MATLAB not only provides a student with a valuable tool for solving common engineering and scientific problems, but also serves as an excellent foundation for learning the more complicated components of high-level computer programming languages such as C. MATLAB offers a simplified programming environment ideal for focusing on and learning the concepts of using and naming variables, file input and output, for and while loops, if and switch/case decision structures, and program modularization with functions.

In C, programs are compiled as a whole and linked with necessary libraries to produce an executable object file which can be run independently from the C development and compiling software. In contrast, MATLAB translates commands line by line. MATLAB uses a command window environment in which single commands can be typed and executed one by one. Larger blocks of statements comprising a program can be typed and saved as a text file and executed from within the command window. Translation and execution of such text, or files occurs line by line.

For students first learning the concepts of programming, an environment which reduces the time spent debugging allows for greater focus on and understanding of the more difficult concepts necessary in algorithm development.

In MATLAB, the student must learn the same conventions for naming, assigning values to, and using variables to develop a program as are necessary in C. However, in contrast to C, all variables in MATLAB are treated as an array, and all values within arrays are stored as double precision real numbers. This eliminates not only having to declare a variable at the beginning of a program, but also having to correctly identify the type of value stored and the dimension and size of arrays. Although these conventions will ultimately be mastered when the student learns C, in MATLAB the student has the opportunity to become familiar with the beginning concepts of variable usage in programming.

The programming structures available in MATLAB - *for* and *while* loops, *if* and *switch/case* - are very similar in configuration to those in C but use somewhat simpler, more straightforward syntax. For example, rather than enclosing the set of statements to be executed by a for loop, while loop, or if-structure within braces, in MATLAB the command “end” is used to complete each structure. Though the differences in these structures are subtle, they lend themselves to more intuitive understanding for the beginning programmer without sacrificing any of the programming capabilities.

File input and output can also be introduced within the MATLAB environment in a more direct way without having to utilize the concept of pointers as is necessary in C. The same is also achieved with the treatment of functions. MATLAB is an excellent platform for teaching program modularization with functions, again without having to incorporate the more difficult concept of pointers. Functions are called and defined very similarly in both MATLAB and C, but in MATLAB, the passing of values into and out of a function is more straightforward. Function arguments are used only for passing values to a function, while values are returned only through output arguments. Unlike functions in C, MATLAB functions can be defined with as many return values as desired. Thus, pointers are not needed to “pass” arrays to and from a function. In MATLAB, an array can be passed to a function as an argument in its entirety, and because the address is not used to pass the array, any operations performed within the functions using the array will be treated locally and will not affect the values of the array sent in from the main program.

The other major teaching tool MATLAB offers is that function calls can be introduced as simple m-files. That is, all the variables can be treated as local variables. You can write a MATLAB main program (file) that defines all the variables and then have the main program call a number of m-files without passing any variables. As long as the variable names in the m-files are the same as in the main program, all variables are treated as local variables. Thus, you can teach function calls in four steps. Step one, the students write a program that performs all the tasks of the program in one m-file. The program should have an input component, a calculation component and an output component. Step two, the students cut and past the various components into separate m-files. Then in step three they can convert the m-files to functions and finally in step four they change the names of the variables in the functions. This allows the instructor to not only teach the concept of local and global variables, but also the concept of variables that are local to each function. This is one of the hardest concepts in C for students to understand. MATLAB offers the instructor a very easy means of teaching this rather complex programming idea. Thus, something that is introduced in EXCEL, discussed in HTML can now be used and explained in MATLAB.

Familiarity with MATLAB does not only provide the student with a valuable tool for solving many common engineering problems encountered both in undergraduate and graduate level curriculum, and in the laboratory environment, but also provides a more friendly environment for learning and becoming proficient with the fundamental concepts of programming. When the student is then faced with learning the C language, the challenge will be towards understanding the higher concepts of memory allocation, pointers, string manipulation, and the programming syntax necessary to link libraries, define variables and functions, and write code.

C

We spend the remaining weeks in ENGR0012 teaching C. During the first week we discuss variable types, the different syntax in the looping and branching, and the input output procedure in C. During week two, we introduce function calls and by week three we are discussing pointers. Thus, all of the material we used to take 16 weeks to teach can now be covered in just three weeks. This leaves us four weeks at the end of the semester to reinforce covered concepts through various programming applications for problem solving.

RESULTS

To measure the success of using these different approaches for teaching C programming, we conducted surveys of the current freshman class (after just completing the MATLAB portion of the second semester Engineering 0012 course) as well as upper class engineering students.

The freshman survey was administered on line to 350 current freshman engineering students. It primarily consisted of asking students to respond via a 1 to 5 scale (from strongly disagree to strongly agree) to statements regarding the effectiveness of teaching and learning in ENGR 11 and 12 as describe in this paper. The percentage of students that responded "agree (4)" or "strongly agree (5)" to the various statements is shown in Table 2.

The upper class survey was administered to 182 students across all engineering disciplines. Many of the statements in the freshman survey were replicated in the upper class survey and additional statements were added to address the effectiveness of teaching C programming. The percentage of students that responded "agree (4)" or "strongly agree (5)" to the various statements is shown in Table 3.

The freshman survey results show a clear perception by the freshman students that support the effectiveness of the teaching style adopted in ENGR 11 and 12. The survey focuses on the programming concepts that we have identified as being particularly difficult for students to learn. With the exception of the connection between EXCEL and UNIX and these concepts, over 80% of the students agreed or strongly agreed that learning other software tools has prepared them to learn the challenging aspects of C programming. The smaller percentage of students agreeing with the connection to concepts in EXCEL may be because EXCEL is taught very early in the freshman year and the survey was administered midway through the second semester. Thus students may not clearly see the concept connections until later in their academic careers. This is also evidenced by the upper class survey results shown in Table 3. Smaller percentage of students agreed or strongly agreed with many of the effectiveness statements. However, 70% of this group of students did agree that the concepts learned ENGR 11 and 12 were helpful in later engineering courses and that these courses are a valuable part of their engineering curriculum.

CONCLUSION

We believe that this new approach to teaching C programming is improving the learning process for the students. By taking the students through a step by step introduction to the various components of programming using different software, we are allowing the students to learn the material in an environment that makes "seeing" the concepts easier.

We have also discovered that the fear factor, although still present in the minds of the students, has been reduced. Students are more willing to try and write the code where before they tended to quit. This is very important, because you cannot learn to program if you are not willing to try.

REFERENCES

[1] Hanly,j.R., Koffman, E.,B., Horvath, J.,C., "C Program Design for Engineers", Addison-Wesley, 1995.

[2] Tan, H.,H., D'Orazio, T.,B., "C Programming for Engineering & Computer Science", McGraw Hill, 1999.

[3] Etter, D.M., "Introduction to ANSI C for Engineers and Scientists", Prentice Hall, 1996.

[4] Ludi, S, Collofello, J., "An Analysis Of The Gap Between The Knowledge And Skills Learned In Academic Software Engineering Course Projects And Those Required In Real Projects", Proceedings *2001 Frontiers in Education Conference,* Oct. 2001, Reno, NV, pp. T2D-8 - T2D-11.

[5] Fincher, S., What are We Doing When We Teach Programming?, Proceedings *1999 Frontiers in Education Conference,* Nov. 1999, San Juan, PR, pp. 12a4-1 - 12a4-5.

[6] Nelson, M.L., Rice, D., "Introduction To Algorithms And Problem Solving", Proceedings *2000 Frontiers in Education Conference,* Oct. 2001, Kansas City, MO, pp. S2C-5.

[7] Raymond, D.R., Welch, D.J., " Integrating Information Technology And Programming In A Freshmen Computer Science Course", Proceedings *2000 Frontiers in Education Conference,* Oct. 2001, Kansas City, MO, pp. T4C-7 - T4C-11.

[8] Jermann, W.,H., " The Freshman Programming Course: A New Direction", Proceedings 1996 ASEE Annual Conference, June 1996, Washington, DC.

[9] Westbrook, D.S., " A Multiparadigm Language Approach to Teaching Principles of Programming Languages", Proceedings *1999 Frontiers in Education Conference,* Nov. 1999, San Juan, PR, pp. 11b3-14 - 11b3-18.

TABLE 2:
RESULTS OF FRESHMAN SURVEY

Survey Statement	**Percentage of freshman students that responded "agree" or "strongly agree"**
Learning EXCEL helped me to understand data manipulation using one and two dimensional arrays.	**70.6**
Using multiple worksheets in EXCEL helped me to understand referencing data between 2 different files in MATLAB.	**49.0**
Learning logic in EXCEL (IF, AND, OR, etc.) helped me understand program branching in MATLAB.	**64.8**
Learning about the UNIX file management system (present working directory, changing directories, etc) helped me understand setting the path in MATLAB.	**61.1**
Learning HTML coding helped me understand the importance of formatting (white space, indenting, etc.) in writing code.	**82.6**
Learning HTML coding helped me to understand the importance of correct syntax and how to debug code.	**88.4**
Learning HTML framesets helped me to understand how one file can reference another file.	**81.2**
The material and concepts I am learning in ENGR 11 and ENGR 12 have helped me to understand how to logically layout a problem.	**82.2**
After finishing with MATLAB (before beginning C), I have a basic understanding of program control using arrays, branching, looping, etc.	**92.8**
The material and concepts I am learning in ENGR 11 and ENGR 12 have improved my problem solving skills.	**79.2**
I believe that the material and concepts I am learning in ENGR 11 and ENGR 12 will prepare me for classes I'll take in my sophomore, junior, and senior years.	**79.6**

TABLE 3:
RESULTS OF UPPER CLASS SURVEY

Survey Statement	Percentage of freshman students that responded “agree” or “strongly agree”
Learning EXCEL helped me to understand data manipulation using one and two dimensional arrays.	**74.8**
Using multiple worksheets in EXCEL helped me to understand referencing data between 2 different files in MATLAB.	**40.9**
Learning logic in EXCEL (IF, AND, OR, etc.) helped me understand program branching in MATLAB.	**50.9**
Learning about the UNIX file management system (present working directory, changing directories, etc) helped me understand setting the path in MATLAB.	**40.4**
Leaning HTML coding helped me understand the importance of formatting (white space, indenting, etc.) in writing code.	**69.9**
Learning HTML coding helped me to understand the importance of correct syntax and how to debug code.	**64.5**
Learning HTML framesets helped me to understand how one file can reference another file.	**55.0**
Learning coding in MATLAB prepared me to write programs in C.	**67.4**
Learning coding in C prepared me to write programs in other languages in my engineering classes.	**36.8**
The material and concepts I learned in ENGR 11 and ENGR 12 helped me to understand how to logically layout an engineering problem.	**61.1**
The material and concepts I learned in ENGR 11 and ENGR 12 have improved my problem solving skills.	**62.6**
I believe that the material and concepts I learned in ENGR 11 and ENGR 12 were helpful in classes taken after my freshman year.	**69.8**
ENGR 11 and ENGR 12 were valuable classes.	**70.4**