FOUR STEPS TO TEACHING C PROGRAMMING

Dan Budny[1], Laura Lund[2], Jeff Vipperman[3] and John L. Patzer II[4]

Abstract *— Our experience with teaching C programming has shown that the students have a problem understanding the concept of arrays, dealing with the sytanx of the language, designing the organization of the program and understanding the concept of flow control such as looping and branching or function calls. In a typical C programming course the instructor must deal with all of these problems simultaneously, because of the nature of the language. To help solve this problem we have divided the various concepts and used different software packages to introduce each topic independently. For example we have discovered that EXCEL can be used to explain the concept of an array, matrix operations, data input, and the built in functions provides the student with a number of useful tools. The concept of designing the layout of a program can be introduced very well with HTML, and then the concept of control such as looping and branching can be introduced with MATLAB without many of the syntax problems that comes with C. Finally once the students are familiar with the use of EXCEL, UNIX, HTML and MATLAB the introduction of C is much easier for the students to understand. This paper describes how we introduced this new teaching concept into the University of Pittsburgh Freshman Engineering computing course.*

Index Terms — Freshman, Programming skills, Problem solving.

INTRODUCTION

A review of most tables of contents of a basic 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

Over the past few years 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 a much more 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 the first semester and instead, the course will 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 separate.

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 strugling with many of the same problems and concerns discussed in the literature. We have found that when teaching freshman the basic concepts that are difficult for them to grasp are difference between the various variable types, arrays,

[1] Dan Budny, University of Pittsburgh, Engineering Student Services Center, B74A Benedum Hall, Pittsburgh, PA 15261 budny@pitt.edu
[2] Laura Lund, University of Pittsburgh, Benedum Hall, Pittsburgh, PA 15261
[3] Jeff Vipperman, University of Pittsburgh, Benedum Hall, Pittsburgh, PA 15261
[4] John L. Patzer, University of Pittsburgh, Benedum Hall, Pittsburgh, PA 15261

0-7803-7444-4/02/$17.00 © 2002 IEEE

functions, pointers and the basic flow control in a program. Then once you think they understand these concepts, you must teach them how to layout the logic of a program. Trying to do all this in 15 weeks is impossible.

OUR SOLUTION

Over the years we have struggled with the traditional approach to teaching C. The basic problem is that all the typical C text books treat the subject basically the same in the order of presenting the material, and use the outline listed in the previous section. However, we have discovered that the problems that the students have with C appear to be centered around the concept of arrays, functions and the basic layout and design of a program. A review of the text books reveals that this material is typically at the end of the semester. Thus, the problem is how do you move this material to the beginning of the semester? Our solution was to design a two semester sequence that uses EXCEL, HTML and Matlab to introduce arrays, functions and program layout design first, then introduce C.

BACKGROUND

Engineering 0011- Introduction to Engineering Analysis

All students are required to take four core courses during their first year. There are two zero-credit seminar courses and two three-credit introductory problem solving courses that are a part of this core. ENGR0011 is a required first semester three credit course, for all freshmen engineers that meets twice a week for 2 hours in a computer-equipped classroom. It is an *integrated* course that has the following overall goals:

To teach the basic computer skills, and their role in problem solving,
To introduce teamwork,
Improve writing and communication skills
To illustrate the use of software packages (MSWord, PowerPoint) in communications.
To begin understanding how material in the basic sciences and mathematics is used by engineers to solve practical problems of interest to society.

The main goal of the course is to solve a given problem, using "Excel" and "Matlab" and to produce the required output and or result. In this first semester the students are introduced to a number of the various concepts and skills required to be successful in engineering, and how to use the above software to solve engineering problems..

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

It is the experience of the faculty that students know very little about the actual operation of a computer or computer software as problem solving tools. The students are good at using AOL instant messenger, 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 lost. Thus, the main focus of ENGR0011 is to begin the process of structured thinking.

Engineering 0012- Engineering Analysis [SPRING]:

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 is also an *integrated* course that 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 how the output of one program can be used as the input into another software package.
- 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, then centers more around the use of "C" to solve the given problem.. ENGR0012 is the next step in the development of the students problem solving skills. In the first semester they 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. Here 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 is Introduction to Arrays, Matrices, Linear Algebra and function calls. During the process of covering this material we also introduce curve fitting (linear, semi-log and log-log), goal seek and a number of matrix operations.

The basic concept we try and show the students is that a spread sheet is nothing more than a matrix or a two dimensional array. Thus, we can teach the concepts of arrays using a very graphical approach with EXCEL. The only disadvantage of EXCEL is 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 the difference to stress the concept of referencing an element within a array.

Since we started teaching EXCEL, we have observed that the vast majority of the students do not understand matrix operations, or have never been exposed to it before. This would help explain why arrays appear so difficult for them to grasp in C. The introduction of matrices in EXCEL is definitely helping the students understand the concept of arrays in both Matlab and C.

Another concept that we can introduce with EXCEL is the concept of 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.

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 difference "workspaces" or a very basic form of a function call. We use the multiple sheets to solve problems. For example the data may be stored on one sheet, but we have them do the calculations on a different sheet. We explain 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 is a very simple and user friendly software that allows us to introduce: arrays, input/output, and functions in a manner that is impossible in C. It is a baby step but we have found 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 concept problem, was that of the "path". Matlab uses m-files to store the various main program and all the functions. To use them the user must set the path to the files. That is the user must understand the concept of file management. Once again we discovered this is not a concept that most freshman understand. The second main concept problem was how to layout the design of a program with branching, looping and function calls.

To help with these concepts, we decided to spend the next four weeks teaching HTML programming before we introduced 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 knew that file management was a problem, we used HTML as an excuse 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 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 found that the layout of a web page allowed us to begin to teach 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 introduces the student to the practice of laying out code line by line. We tell the students that the reason we are teaching web page design is not 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. These simple baby steps are designed to target problem areas we have observed when teaching a traditional C course.

We stress the proper use of comment lines 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 can begin the process of learning to be detailed in the writing of a code. In fact, by requiring the students to design their web pages using tables, you can reinforce the need for watching 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 mistake.

HTML also offers the instructor the opportunity to introduce function calls, by converting a web page into a frame set. Basically 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. This concept allows you 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.

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 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). It was initially developed as a tool for performing matrix operations, but has evolved into a popular interactive system and programming language with numerous available Toolboxes, or subroutines, for common problems encountered in any of the various engineering and scientific disciplines. We spend the last five to six weeks of the first semester ENGR0011 (we also spend a week on Word and PowerPoint and there is also a week lost in testing) and the first five 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 is referred to as an interpreter which 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 script, files occurs line by line.

Though execution of programs with an interpreter is inherently less efficient and is dependent on the interpreter software to run, the benefit is a substantial reduction in time spent debugging compilation errors. Furthermore, script files executed within the Matlab environment have access to all Matlab functions so that specific header statements identifying necessary libraries are not required. 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 fashion without having yet 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 from a function is more straightforward. Function arguments are used only for passing values *in* 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 first be introduced as simple m-files. That is all the variables can be treated as local variables. You can write a Matlab main program that defines all the variables and then have the main just call a number of m-files without passing any variables. So long as the variable names in the m-files are the same as in the main, all variables are basically treated as local variables. Thus, you can teach function calls in four steps. Step one have the student write a program that performs all the task of the program in one m-file. The program should have an input component, a calculation component and an output component. Step two have the student just cut and past the various components into separate m-files. Then in Step three have them convert the m-files to functions, finally in step four have them

change the names of the variables in the functions. This allows the instructor to 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 the 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 will 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 will also provide 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

Finally we spend four weeks in ENGR0012 teaching C. We spend one week discussing variable types, the different syntax in the looping and branching, and the input output procedure in C. Week two is function calls and week three we are discussing pointers. Basically we teach all the material it use to take us 16 weeks to teach we now cover in four to five weeks. This leaves us four weeks at the end of the semester that we use to do nothing but solve problems.

With this new course design we are spending all the time in C discussing functions and pointers. The design projects we give takes the students to depths of programming that we were never able to achieve before.

CONCLUSION

We believe this new procedure to teaching C programming is improving the learning 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 software environment that makes "seeing" the concepts easier for the students.

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 can not 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.