A NEW APPROACH TO TEACHING C PROGRAMMING

T. Grandon Gill, IS & DS Department, CIS1040, University of South Florida, Tampa, FL 33620
E-mail: ggill@coba.usf.edu, Phone: 813-974-6755, Fax: 813-974-6749

ABSTRACT

Although the C programming language is an excellent language for teaching computer concepts, it is a difficult language for MIS majors to learn. The paper presents a new approach to teaching C that is project oriented, makes extensive use of multimedia and online discussion technologies, and is highly flexible in terms of delivery, with in-class, distance learning and hybrid delivery approaches all having been applied successfully. The techniques employed in the class—including those for content creation, hybrid delivery and assignment validation—should be readily adaptable to other subjects, particularly those with high levels of technical content.

MIS, Programming, Distance Learning, Multimedia, Pedagogy

THE PROBLEM

Teaching C as an introductory programming language provides an instructor with a unique ability to mix instruction in programming with other topics. The “low-level” nature of C makes it particularly well-suited for presenting issues such as data representation, memory organization and I/O systems—particularly useful in MIS curricula lacking separate courses covering such topics. There are, however, also a host of disadvantages of using C to introduce programming: its awkward syntax, its teletype-style I/O and the widespread perception (among students) that it is a “dead” language. The problem, then, is to teach C in a manner that realizes its pedagogical benefits in teaching systems concepts without falling prey to its deficiencies as a tool for teaching beginners how to program.

LEVEL OF STUDENTS

Undergraduate MIS majors, taking the class as a first programming class.

NUMBER OF STUDENTS

Presently, between 80 and 120 students per semester.

MAJOR EDUCATIONAL OBJECTIVES

- To engage students in the programming process, for consideration as a potential career path
- To acquaint students with computer concepts they will not learn in other MIS courses
- To instill sufficient programming skills for success in the remainder of the MIS major
- To familiarize students with the tools and techniques of modern software development

INNOVATIVE AND UNIQUE FEATURES

There are a variety of tools and techniques that are integrated into the course that make it quite unique. These include:

A. *Use of exercise-enhancing software tools that were designed and programmed for the course by the instructor.* For example, *FlowC* (a flow charting tool that also generates C code) is used to introduce "generic" programming constructs and is then used to produce C source code that can be imported into Visual Studio .Net projects while students are becoming comfortable with C's awkward syntax. *ServerSim* (a browser-based application that simulates a web server) allows students to create CGI programs as a final course exercise, instilling the perception that C can be used for something useful and relevant. *WinConvert* & *WinBitwise* allow students to specify integer conversion and bitwise operator problems and then display the entire process used to derive the solution—rather than merely displaying the solution itself.

B. Use of over 40 narrated .avi files developed by the instructor and distributed to students on CD. The use of these files—mainly animated screen captures lasting 5-40 minutes apiece—allows students to review materials that could easily be missed in lectures, replay sections as needed, and view tool walkthroughs while performing the same task using the tool being demonstrated. They also serve as a replacement for about 50% of normal lecture time.

C. Elimination of traditional exams. Seven exercises (5 involving programming) are used in lieu of exams in determining grades. In order to get credit for most assignments, however, students must pass a "validating" exam specifically targeted to the assignment after turning it in. For the three largest programming assignments, oral exams on the code prepared are administered by the instructor or, in some cases, teaching assistants (TAs). For three other assignments, validation is accomplished using computer-administered exams. Two of these validating exams are created using automated test-generation tools developed by the instructor—making it possible to generate nearly limitless numbers of exams at negligible cost.

D. Tight integration with a web-based course delivery tool. Online discussion group support of assignments is central to the course approach, with single assignment discussions frequently exceeding 100 postings. In addition, the course delivery system is used to administer validating exams, provide access to streaming versions of weekly lectures and for delivering course announcements and materials.

E. Focus on using tools, especially for debugging. Unlike many C courses—which emphasize lack of tool specificity as a virtue—the active learning philosophy of our design leads us to embrace effective use of our chosen tool (Visual Studio .Net). As a consequence, two of our assignments are essentially tool walkthroughs, requiring the student to first observe (in .avi form), then recreate, effective programming and debugging techniques.

F. Flexible classroom/web-based delivery: All course content is made available online, making it possible for students to choose to attend in-class lectures, or view lectures prepared for the web, or both.

COURSE CONTENT

The material being covered in the course is fairly comparable to what is taught in a "normal" introductory programming course, with some important exceptions: 1) greater emphasis on the conceptual nature of programs and data, 2) greater emphasis on pointers, structures and layout of memory, 3) greater emphasis on the specific programming environment we have chosen (Visual Studio .,Net). (1) and (2) stem from our desire to introduce students to computer organization and architecture, in addition to pure programming. (3) is the result of the instructor's past experience: students who are comfortable with the development tools by the middle of the semester tend to be able to handle projects that are much more substantial by the end of the semester—since good debugging tools and techniques dramatically increase a student's ability to deal with code complexity. A full week, near the end of the semester, is also spent introducing HTML, web forms and CGI processes. This feeds into the final assignment, in which the students create a CGI program in C that processes a web form and responds with a web page.

COURSE ORGANIZATION

The best way to characterize the course organization is assignment-centric. Because a student's entire grade will depend on assignment grades, all lecture content is specifically presented in terms of how it relates to one or more assignments. The organization of the Summer 2003 course assignments is as follows:

1. *Simple project:* Students create a single file project, a multi-file project and perform various debugging activities. Their activities essentially recreate a demonstration .avi file they are given.
2. *Numbering systems:* Students perform base conversions, hex arithmetic, bitwise operations and twos-complement conversions. Validated by a computer-generated online quiz.
3. *Flow Charting:* Student use the FlowC tool developed for the course to create a series of flow charts, starting with the very qualitative (e.g., diagnosing a printer problem) and moving to the algorithmic (e.g., computing a mortgage payment using successive approximation). Later flow charts are intended to be C-compliant, so students generate

them as C projects then compile them in Visual Studio .Net. Validated by oral exam with the instructor or teaching assistants.

4. *Debugging:* Students are first given code for an application with known bugs and are required to parallel an .avi file that walks them through the process of bug identification and removal. They are then given the same basic code with a different set of bugs and are required to identify and eliminate the bugs on their own. Validated by having the students repeat the second half of the assignment in a lab setting.
5. *Memory Grid:* Students identify data and addresses in a simulated block of memory populated by primitive data elements, pointers and structures. Validated by a computer-generated online quiz.
6. *Functions:* Students create a set of eleven C functions, emphasizing string manipulation, and use a test script to verify their results. Validated by oral exam.
7. *CGI Program:* Students create a CGI program that responds to web form (specifying loan amount, term and interest rate) with a mortgage amortization table. Students run the program under ServerSim, a program that simulates a web server but runs locally. Validated by oral exam.

PRESENTATION

Extensive materials have been prepared to support the new approach to teaching C programming. Most are delivered to students on a CD distributed by the school bookstore (or students may make their own copy). On the CD are included:

- A draft textbook (665 pages) delivered in Acrobat (.pdf) format, written by the instructor. Hard copy is also available for sale at a local copy store.
- .avi files (41 in all) covering specific topics integrated with the draft textbook.
- Course lecture notes (.pdf, ~237 pages) containing all the slides displayed in lectures. These are also used to prepare lecture .avi files, created weekly
- Assignments (.pdf, 46 pages)
- Course software, including the FlowC, ServerSim, Winconvert and Winbitwise applications, plus running versions of all assignments and source code used in demonstrations.
- The same content—excepting the .avi files, which are too large—is also provided on the web-based course delivery system (Blackboard). Some content is limited to the delivery system, including:
    - Weekly lectures (.avi and .wmv versions of the weekly lectures, created specifically for the web)
    - Discussion groups, one for each assignment plus groups for general questions and organizational groups (e.g., for scheduling oral exams).

- Online quizzes. Assignment quizzes, generated automatically using instructor-developed software. Practice versions of the exams are also available for students.

EFFECTIVENESS AND SPECIFIC BENEFITS

There are a number of quantitative indicators that suggest the pedagogy we have adopted for the course has been effective in its objective of engaging students in their programming activities. Because the course (and pedagogy) has evolved over the past three semesters, one way to look at this is longitudinally, as shown in Table 1.

Table 1: Measures of Course Effectiveness

	Fall 2001	Spring 2002	Fall 2002
Description	First time taught. Elements present: • Some .avi files • 6/7 exercises comparable • No course delivery system • 2.5 hours lecture/week	Elements introduced: • FlowC & new Assignment 3 • Blackboard support	Elements introduced: • Draft text used • Complete .avi support • 1.25 hours lecture per week • 1.25 hours optional lab per week • Taped lectures used for 2/3 sections
Grade Distribution			
Percent Passing	90%	90%	84%
Percent "A" grades	23%	43%	63%
Course Evaluations (1-5, where 5 is best)			
Percent filling out course evaluations	49%	44%	23%
Communication of Ideas and Information	2.55	4.35	3.21
Overall Instructor Rating	2.63	4.47	3.76
Online Discussion Tools			
Postings/student	N/A	12	13
Median Est. Accesses/week	N/A	2	5.7
Est. % Accessing 2X per week or more	N/A	50%	84%

The most significant measure—from the standpoint of meeting course objectives—is probably the percentage of A grades awarded. This percentage is virtually a proxy for percentage of students completing all assignments. Because the two assignments most likely to be left incomplete (6 and 7) remained unchanged throughout the 3-semester period, and because consistent performance was confirmed by oral exams in all three semesters, the rise from 23% to 63% represents a major achievement.

Student evaluation measures, on the other hand, have proven to be unhelpful. The biggest problems are a consequence of sampling issues. Specifically, the very low sample sizes reflect the fact that evaluation instruments were administered in a final exam review session—the last class meeting. By this time (particularly in the Fall 2002 semester), the majority of students had already met the requirements for an A by passing their oral exams on assignments 3, 6 and 7—or felt certain they could do so. Thus, they skipped the review session. The result: a sample of students in attendance consisting, almost entirely, of students who felt—or knew—they would need to take the final exam in order to pass the course. This was not an ideal sample, from the instructor's perspective. To address this issue, in Spring 2003 the instructor sought, and was granted, permission to depart from normal university procedures and to administer course evaluations online, as well as in class.

Another important measure of “success” is IT usage. For Spring and Fall 2002, usage measures from the course delivery system are available, summarized in Table 1. The results show both substantial usage (6 times/week in Fall 2003) and growth in usage (nearly tripling from spring to fall). A significant (p-value of ~0.02) positive relationship between level of system usage and course grade was also observed. This is made more meaningful by the fact that all usage of the course delivery tool was (and is) optional, and students are given no special incentives for online participation.

TRANSFERABILITY

The question of transferability can be address in two ways, 1) relating to teaching C specifically and 2) related to teaching, in general. With respect to teaching C, the instructor is currently developing a more polished version of the draft textbook he has been using, incorporating all the software, .avi files and teaching techniques developed for the course. As a result, it should ultimately be possible for instructors to employ these tools and techniques to teach C, should they desire to do so.

On a more general level, through data collection to be conducted over the next year, we intend to develop a more rigorous sense of the educational efficacy of three techniques central to the curriculum design:

- The use of “validated” exercises as a substitute for traditional examinations.
- The use of focused .avi clips to supplement (and partially substitute for) full length lectures
- The effectiveness of the hybrid distance learning/classroom course model

Unlike the C-specific tools designed for the course, none of these three areas are content-specific. We therefore hope that the results of the analysis—combined with the practical lessons learned in actually teaching using the techniques—will encourage others to use the pedagogy described here.