Teaching in Shifting Sands: Changes in CS2

Richard Close
United States Coast Guard Academy
New London, CT
rclose@cga.uscg.mil

Danny Kopec
Brooklyn College
Brooklyn, NY
kopec@sci.brooklyn.cuny.edu

Jim Aman
Columbus School for Girls
Columbus, OH
jimaman@acm.org

1. Abstract

At a time when the Curriculum 2001 committee is compiling their recommendations, it is perhaps useful to look to the past and track the evolution of the CS2 course. As the computer science curriculum has evolved through the years [4], more has been expected of students in lower-level courses. This may be seen as a natural evolution; a sign of the maturity of the field. Changes in the curriculum, particularly the breadth-first approach, have affected the content and delivery of every course. The net result is that the first course in computer science is less concerned with teaching programming skills in a single language, a burden now shifted to later courses. The CS2 course has become more important in its effect on a student's experience, knowledge, and outlook on the entire computer science major.

One of the most difficult questions for faculty is what precisely should be included in CS2? Should we play to the audience and teach what a fairly large number of students seem to want (Java, web-programming, Visual BASIC, etc.)? Or should we continue with the latest ACM/IEEE recommendations and risk becoming extinct or irrelevant? This paper examines the reactions of three very different institutions with very different constituencies to these questions.

2. Experience at Three Institutions: The Problem in Different Settings

2.1 The United States Coast Guard Academy

CS2 is a course, which has always been a moving target. CS2 was first introduced at the U. S. Coast Guard Academy under the title *Technical Programming* almost 25 years ago. At that time, it had been noticed that computer science majors really did not become proficient programmers in CS1 so a more thorough grounding in programming concepts was offered along with a good dose of theory - data structures and some algorithm analysis. Informally, the course was often known as "Baby Data Structures." This indicated that much of the material would be repeated - albeit at a more sophisticated level - in a subsequent course. The breadth of the first or “foundations” course has, we think, been an important reason why students often enter CS2 with a deficiency in programming knowledge and experience.

The original language of the CS2 course was FORTRAN, which was replaced by Pascal in the mid-80's. By the early 1990's C had been selected to replace Pascal. The prevalent feeling was that real programmers didn't use Pascal and, in any case, a computer science major should know more than one programming language. C turned out to be a little harder for students than Pascal, so some of the theoretical material was dropped. The pervasive fascination with object-oriented concepts, which continued through the 1990's, suggested that it would be an easy transition to C++. Again, the transition was harder than originally forecast and the theoretical topics further diminished. Lately, Java and Visual BASIC are gaining favor. The choice of a popular programming language has been credited with making CS2 more attractive to non-majors, particularly engineers and majors in the information sciences. However, reports from instructors in such courses indicate that the theoretical computer science content is almost gone. In many ways, CS2 has become a programming course akin to what CS1 once was. This possibly relates to the popularity of the supposedly practical rather than theoretical aspects of the discipline. Many students (and faculty members) recognize that knowing Java and/or Visual BASIC may make them more employable. Even in the Coast Guard, we are feeling the force of a booming marketplace on our curricular decisions.

2.2 Wilmington College, Wilmington, Ohio

Unlike the U.S. Coast Guard Academy or Brooklyn College, the student body at Wilmington College is a heterogeneous mixture of students from metropolitan and agricultural areas. The cultural mix is evident both socially and educationally. Computer science classes reflect very different preparation and experience as students from Cincinnati, Dayton, and Columbus work alongside students from small, rural school districts and more used to helping on the family farm than dashing off to the shopping mall.

The CS2 course at Wilmington College is an intensive study of a high-level programming language. C++ has been recently introduced after several years with Pascal. Besides the obvious need for instruction in the specifics of the language, this course carries a strong emphasis on fundamental algorithms and basic software engineering design techniques. Study of algorithm development is continued from the strong basis constructed in the CS1 course**.**

Of particular concern for the design of CS2 is the selection of the most appropriate textbook – which is no trivial task. Unlike CS1, there are no clear language independent candidates. Most texts are carefully designed and well written, but they generally lack a particular characteristic, which in our experience has been difficult to find: cohesion among problem sets. One example of a Pascal-based text we found which advocates the case study approach is *Designing Pascal Solutions: Case Studies with Data Structures* [3]. Local experience over the past decade suggests that students make more effective conceptual leaps when the programming assignments represent progressive refinements to a few problems.

The pervasive problem with the quest for effective case studies has been striking a balance between good (or acceptable) problem sets and implementation-appropriateness. In our case this latter term means having a book which deals with Turbo Pascal for Windows, rather than just Pascal. The book also needs to place emphasis on the particular set of issues, techniques, and skills we consider most appropriate. The search is never perfect, but it continues**.**

2.3 Brooklyn College, Brooklyn, New York

In the introductory programming course (CS1), students learn the foundations of programming through sound, structured, top-down techniques. But in CS2 *(CIS 15: Advanced Programming Techniques Using C*) issues of how to best proceed with computer science programming instruction arise.

Bigger, modular problems illustrating various themes fundamental to the design and use of functional programming are posed. In addition to the Unix operating system, there are critical issues centered around such topics as multi-file programs, data representation and conversion, program storage structures, parameter passing, scope and recursion, internal representation of elementary data structures and abstract data types. Along the way the elementary data structures (stacks, queues, and linked lists) as well as typical searching and sorting techniques may be covered. The course is completed with the study of pointers and file structures**.**

In CS1 and CS2 many instructors will assign diverse programming work of a theoretical nature. Such assignments may do well to illustrate the difficulties for a particular language to handle certain I/O constructs or to perform (or implement) operations on certain data structures. For example, in CS2 students may be asked to handle character manipulation or to perform operations on arrays, stacks, or queues.

3. Lessons Learned

It seems that CS2 instructors should concentrate more on presenting programming problems of a more practical value and of natural interest to students. This definitely would include problems which need solutions to be programmed, as opposed to using one of the many applications available today (for example, spreadsheets, databases, or standard financial and business packages). The assignments should be selected carefully to illustrate the benefits derived from modular and object-oriented approaches. Students' efforts in solving such problems will accomplish many important goals:

1) They will derive personal satisfaction in using what is learned in the classroom for practical application.
2) They will develop a hands-on insight for the issues involved in problem solving.
3) They will be able to consider the tradeoffs between a variety of possible programming approaches and intelligently choose from amongst them.
4) They will be confronted with and will have to learn how to deal with language-specific issues relevant to their programming problems.

Specific examples of assigned programming projects are available directly from the authors. Some typical problems that have been used successfully are (1) a general purpose currency exchange program; (2) a functional program which draws figures based on specification of rectangles; (3) a simulation of an airline reservation program and (4) simulation of a popular, knowledge-based TV quiz show. Note that these vary widely in difficulty. Some are potentially weekly homework exercises while the simulations could be semester projects.

While content is of central importance to a course, some mention of new and novel delivery schemes should be made. Currently, many courses have web sites associated with them. Beyond these static pages, however, there are some web-based and other tools, which could revolutionize computer science instruction [2]. For example, a system called "WebToTeach" has been developed for a variety of computer science courses including CS2. This system offers a web-based interactive programming environment. It employs automatic program-checking software and is designed to be extremely easy to use for faculty and students. Another feature is that it encourages sharing of exercises or parts of exercises among faculty (and students). Instructors are able to specify programming problems and the acceptable solutions to them. Hints are supplied and solutions can be fully accepted, partially accepted, or rejected. The system is available to anyone with web access and is being used and tested at a number of academic institutions. During the Winter-Spring 2000 semester, it was tested for comparison purposes in two sections of CS2 taught by the same instructor at Brooklyn College. A complete evaluation of this pilot study has not been completed at this point but preliminary results are encouraging.

Another important problem is to address the retention of students in the computer science major. CS2 will often be the course where students will decide whether to stay in the major or seek other "more facile majors." While content and delivery are of utmost importance, it could be that a novel programming environment could help persuade students to persevere. Although a system like WebToTeach does not represent a full intelligent tutoring system for teaching computer science, it does appear to be a step in the right direction. In the future we might envisage a system which would embody deep knowledge about its subject matter, which would be able to construct a model of the learner, and which would have tutoring expertise, coupled with multimedia presentation skills. But maybe that's overkill for CS2.

4. Acknowledgement

The authors wish to acknowledge Dr. Nosan Yanowsky of Brooklyn College for his assistance in the preparation of this paper.

5. References

[1] Aman, J., Close, D., and Kopec, D. (1999) Panel presentation: "How Should Data Structures and Algorithms Be Taught?" In *Proceedings of the Conference on Innovation and Technology in Computer Science Education*, ITiCSE'99, Krakow, Poland.

[2] Arnow, D. and Barshay, O. (1999) WebToTeach: A Web-based Automated Program Checker. *Frontiers in Education* (FIE '99), San Juan, Puerto Rico (Nov., 1999).

[3] Clancy, W. and Linn, M., (1996) *Designing Pascal solutions: Case studies with data structures*. W.H. Freeman and Company, NY.

[4] Tucker, A. (ed.), Barnes, B., Aiken, R., Barker, K., Bruce, K., Cain, J., Conry, S., Engel, G., Epstein, R., Lidtke, D., Mulder, M., Rogers, J., Spafford, E., and Turner, A. (1991). *Computing Curricula 1991*, ACM/IEEE-CS Joint Curriculum Task Force, ACM Press and IEEE-CS Press, *New York.*