Teaching the Second Computer Science Course:

Lessons Learned and Directions for the New Millenium

Danny Kopec

Department of Computer and Information Science, Brooklyn College

Jim Aman

Director of Information Services, Columbus School for Girls

Dick Close

Department of Science, United States Coast Guard Academy

As we have seen the computer science curriculum evolve through the years (Tucker et.al. 1991), more has been expected of students in lower-level courses. 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. That burden has been shifted to later courses in the curriculum. The second course in the curriculum (CS2) becomes more important in terms of how it will affect a student's experience, knowledge, and outlook on the entire computer science major.

One of the most difficult questions to answer is the decision as to 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, etc.)? Or should we continue with the latest ACM/IEEE recommendations and risk becoming extinct?

Keywords : Computer Science Education, Second Computer Science Course (CS2), WebToTeach, Computer Programming Languages, C, C++, CASE-based approach, Programming Projects.

Experience at Three Institutions

The United States Coast Guard Academy (New London, Connecticut)

CS2 is a course that has always been changing; a moving target. CS2 was first introduced at the U. S. Coast Guard Academy under the title of Technical Programming almost 20 years ago. At that time, it had been noticed that CS 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 often was 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 for why programmers enter CS2 with less programming knowledge and experience.

The original language used in the CS2 course taught in the mid-1980's at the U.S. Coast Guard Academy was Pascal. By the early 1990's C replaced Pascal because the prevalent feeling was that real programmers didn't use Pascal and in any case, a CS 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 pervading fascination with object-oriented concepts which followed in the 1990's suggested that it would be an easy transition to C++. Again, the transition was harder than it originally appeared and the theoretical basis for teaching programming further diminished. Lately Java or Visual Basic seems to be gaining favor. This choice of programming language has made attributed to 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. 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.

Wilmington College (Wilmington, Ohio)

The CS2 course at Wilmington College is an intensive study of a high-level programming language. For several years Pascal was used in this course and will probably continue to be used for the foreseeable future. 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 a 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 we are looking for a particular characteristic which in our experience has been difficult to find: cohesion among problem sets.

One example of a text we have found which advocates the case study approach is *Designing Pascal Solutions: case studies with data structures* (Clancy and Linn, 1996). Experience over the past decade tells us that students make more effective conceptual leaps when the programming assignments represent progressive refinements to a few problems which are built upon.

The pervading problem with the quest for effective case studies to build upon has been striking a balance between good (or acceptable) problem sets and implementation-appropriateness. In our case this latter term means having a book that 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.

Brooklyn College, Brooklyn, New York

After the introductory programming course, (CS1), where students learn the foundations of programming through sound, structured, top-down techniques, 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, typical critical are issues centered around the topics: 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.

Lessons for the New Millenium

It is time for CS2 instructors to concentrate more on presenting programming problems that are more of a practical value and of natural interest to students. This would include problems which need solutions to be programmed, as opposed to using one of thousands of applications available today – for example spreadsheets, databases, or standard financial and business packages, and which may benefit from modular solutions. Students will thereby better understand the purpose and value of being able to program effectively and efficiently. Students' efforts in solving such problems will accomplish many important goals:

1. they will provide personal satisfaction in using what is learned in the classroom for personal convenience;
2. students will develop a hands-on insight for the issues involved in problem solving;
3. students will be able to consider the tradeoffs between a variety of possible programming approaches and intelligently choose from amongst them; and
4. students 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 will be presented in detail, but here we just mention a couple: (1) a general purpose currency exchange program which can be of genuine practical; (2) a functional program which draws figures based on specification of rectangles (figures may then be scaled up or down, and rectangles may be added or deleted from the figure); (3) a simulation of an airline reservation program; and (4) simulation of a popular, knowledge-based TV quiz show.

In education the value of a new and significant concept or methodology can easily be appreciated. Recently, a colleague, David Arnow, and his research team have developed a series of web-based tools which may help to revolutionize computer science instruction (Arnow and Barshay, 1999). The system called "WebToTeach" has been developed for a variety of computer science courses including CS2. The purpose of the system is to offer a web-based interactive programming exercise system. The program is based on "automatic-program checking software" and is designed to be easy to use for faculty and students. Another aspect of the philosophy behind the system is to encourage sharing of exercises among faculty. Instructors are able to specify programming problems and acceptable solutions to them. Hints can be supplied. Solutions can be fully accepted, or partially accepted, or rejected. The system is available to anyone anywhere with web-based access. It is being used and tested at a number of academic institutions.

An important aspect of WebToTeach is to address the problem of retention of students in the computer science education. In our department CS2 will often be the course where students will decide whether to stay in the major or seek other "more facile majors." WebToTeach does not represent a full intelligent tutoring system for teaching computer science; *i.e.*, 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. Nonetheless, it is an important step in the right direction for computer science education. During the Winter-Spring 2000 semester it will be experimentally tested for comparison purposes in two sections of CS2 taught by one instructor.

Acknowledgement

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

References

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.

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

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

Close, D., and Kopec, D. (1999) Panel Presentation: "How Should The Second Computer Science Course (CS2) Be Taught?" In *Proceedings of the Fourth*

Annual Consortium for Computing in Small Colleges: Northeastern Conference; Providence, RI.

Kopec, D., and Thompson, R.B. (1992). *Artificial intelligence and intelligent tutoring systems: Knowledge-based systems for learning and teaching.* Ellis Horwood Publishers, Chichester, England.

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.*

Authors

Danny Kopec

Department of Computer and Information Science, Brooklyn College

2900 Bedford Avenue

Brooklyn, NY 11210

Telephone: (718) - 951 - 5578; email: kopec@sci.brooklyn.cuny.edu

Jim Aman

Director of Information Services

Columbus School for Girls

Columbus, OH

Dick Close

Department of Science

United States Coast Guard Academy