THE ROLE OF THE DATA STRUCTURES COURSE IN THE COMPUTING CURRICULUM*

PANEL DISCUSSION

Donald Chinn
Computing and Software Systems
University of Washington, Tacoma
Tacoma, WA 98402-3100
dchinn@u.washington.edu

Phil Prins
Department of Computer Science
Seattle Pacific University
Seattle, WA 98119
pprins@spu.edu

Josh Tenenberg
Computing and Software Systems
University of Washington, Tacoma
Tacoma, WA 98402-3100
jtenenberg@u.washington.edu A

1. SUMMARY

What is the role of the data structures course in the curriculum?

Specifically, (1) What is the purpose of CS 103? (2) What is the role of frameworks (if any) in CS 103? (3) Which of the following topics belong in CS 103: Big-Oh, algorithm analysis, recursion, lists, stacks, queues, trees, binary search trees, balanced BSTs, graphs, sorting algorithms, recurrence relations, programming in the medium (as opposed to programming in the large), frameworks, debugging and testing, software engineering?

Tenenberg [T03] "advocates the incorporation of object oriented framework libraries such as the Standard Template Library (STL) into the first data structures course at the university level." In a typical curriculum, data structures is the last of a sequence of programming oriented courses before a fanout of domain specific courses, such as databases, networks, architecture,

---

algorithms, etc. Consequently, those skills absolutely needed in later parts of the curriculum must be placed in the data structures, and the use of many collaborating data structures as part of a medium-sized design is such a skill.

We recognize that there are many variables involved in deciding what topics are discussed in CS 103, such as the academic calendar (quarters vs. semesters), the prerequisite structure, and goals of the program. Even the expertise of the instructor will have an effect on what is taught.

2. DONALD CHINN

Data structures course can be viewed as a transition course. The primary focus of CS 101 and CS 102 is for students to acquire the ability to translate a problem statement into an algorithm and then translate that algorithm into working code. A student who takes CS 103 should not only acquire a larger repertoire of programming tools, but should also acquire the skill and expertise to make thoughtful choices about implementations, which will not only influence program extensibility and program elegance, but also performance. The use of frameworks is not inconsistent with this view, but there is a constraint of time. To fully appreciate the impact of implementation and choice of data structures on performance, students must be actively involved understanding the inner workings of the basic data structures and in the analysis of them, including an understanding of how different inputs can cause different performance behaviors. In a ten-week course, studying the use of frameworks and appreciating their use would be, at best, an incomplete and possibly inefficient use of time, given the other many demands placed on CS 103.

3. PHIL PRINS

The data structures course serves not only the Computer Science major but also Computer Science minors as well as Computer Engineering students and Electrical Engineering students. Hence, we must not only look at the course as a transition from problem solving and basic programming skills to more advanced topics in computer science but also as the final programming course for many students. As such, the course should focus on good programming practices and on the application of data structures to real-world problems. For many engineering students, the performance of their programs will be of utmost importance. On the other hand, this course serves as the computer science student's introduction to topics more complex than merely basic programming. To meet these goals, I feel that this course must give a solid foundation in the theory of internal memory management, provide the student with knowledge of the organization of memory into data structures, and reinforce that knowledge with implementation practice. Performance issues should be considered when introducing the various data structures and their algorithms. The traditional study of the implementation and theoretical cost/benefit of queues under one implementation scheme or another, lists in all of their various forms, and esoteric trees is of secondary importance.

4. JOSH TENENBERG

Novice students should first be taught to understand the use of data structures in practical settings before being taught the construction of data structures from first principles. The data structures studied by novice programming students encapsulate accumulated social knowledge for the storage, retrieval, access, and ordering of large data sets on modern computing architectures. Standardized, extensible framework libraries such as STL serve as the best current way of actualizing this sequencing of topics (use before construction), since they serve as shared implementations of data structures within particular programming communities.

Would we expect young students to understand Russell's Principia before learning to use numbers and arithmetic for practical tasks? Would we expect a hammer or drill maker to design these tools without first understanding the ways in which the carpenter uses the tool in practical settings, the subtle and complex interaction of human, artifact, and environment? How can students appreciate the engineering tradeoffs, forged via the interaction of computer scientists over many years, embedded within the common data structures without first understanding the demands made on these programming tools in building real software? Use before construction is the approach that we take with other complex software systems such as database management systems, operating systems, and compilers; why should we do otherwise with data structures? The current way in which we teach data structures - construction before use - has more to do with implicit and entrenched notions of what constitutes membership in the CS Tribe, what Lister calls "the geekish enthrallment with the machine" [CLTW03], than the pedagogical needs of novice programmers struggling to develop programming skills.

REFERENCES

[CLTW03] Collins, W., Lister, R., Tenenberg, J., and Westbrook, S. The Role for Frameworks Libraries in CS2. SIGCSE 2003, pp. 403-404.

[T03] Tenenberg, J. A Framework Approach to Teaching Data Structures. SIGCSE 2003, pp. 210-214.