4-1992

Course Specifications for New Introductory Courses: Computer Science 227X and 228X

Albert L. Baker
Iowa State University

David Fernández-Baca
Iowa State University, fernande@iastate.edu

Gary T. Leavens
Iowa State University

Course Specifications for New Introductory Courses: Computer Science 227X and 228X

Albert L. Baker, David Fernandez-Baca and Gary T. Leavens
TR #92-09
April, 1992

Department of Computer Science
226 Atanasoff Hall
Iowa Sate University
Ames, Iowa 50011-1040, USA

Course Specifications for New Introductory Courses: Computer Science 227X and 228X

Albert L. Baker David Fernandez-Baca
Gary T. Leavens
Department of Computer Science, Iowa State University
Ames, IA, 50011-1040 USA

April 15, 1992

Abstract

Computer Science 227X introduces first-year students to programming using Scheme. Computer Science 228X is an introduction to data structures in C++. Both courses attempt to teach principles through extensive practice in programming.

This document specifies the course's general objectives and gives an overview of what would be taught.

1 Executive Summary

To replace the existing sequence of introductory courses, CS 211 and 212, the faculty have already approved the creation of four courses: CS 207X, 208X, 227X, and 228X. The 207X and 208X courses are called "C programming I and II" and are designed for non-majors; the 227X and 228X courses are called "Introduction to Programming" and "Introduction to Data Structures." The overall rationale for the change is to service demand for C programming (in 207X and 208X) and also to us to teach our majors as we see fit (in 227X and 228X).

We will to teach 227X as a three-credit course, primarily using the book *Scheme and the Art of Programming*, by Springer and Friedman (MIT Press and McGraw-Hill, 1989).

We will teach 228X as a four-credit course using textbooks still to be decided upon. There are two possibilities that we know of: a book by Riley and another by Budd, both using C++ to introduce data structures. We have made 228X a four credit course, on the recommendation of the faculty, so that the transition to C++ may occur during 228X.

The faculty approved the plans set forth in this document at its meeting on April 8, 1992. Significant changes from past practice and earlier proposals were as follows.

- CS 228X is a four (4) credit course instead of three credits like the old 212. CS 227X would still be a three (3) credit course.
- Math 165 (calculus) is a co-requisite for 208X and 228X instead of a pre-requisite.

- The material on C++ formerly proposed for 227X be taught in 228X instead, so that 227X would be taught entirely in Scheme. This meant not allowing 227X as a prerequisite for 208X.

2 Details on 227X

2.1 Introduction

Computer Science 227X is a course that teaches beginning computer programming. The discipline of computer programming seeks to answer the following questions:

- What are good ways to solve problems with a computer?
- How can one arrive at a good solution surely and quickly?
- What are the costs of various programs? How can they be minimized?

CS 227X addresses all of these questions, although the third question would only be discussed in passing, as it is the province of algorithm analysis courses.

2.2 Course Description

The catalog description of the course is as follows:

> An introduction to computer programming. Symbolic and numerical computation. Recursion and iteration. Modularity and data abstraction. Functional and interactive programming. Imperative programming. Emphasis on principles of programming and program design through extensive practice in writing, running, and reasoning about programs. This course is designed for majors. (3 credits).

2.3 Prerequisites

The formal prerequisite in the ISU catalog will read "2 years of high school algebra and 1 year of high school geometry."

These are intended to ensure a minimal level of mathematical maturity.

2.4 General Objectives

Note: it is standard practice to *not* tell students about affective objectives, such as getting students excited about computer science. So these objectives are not all for consumption by students.

2.4.1 Essential Objectives

In general terms the essential objectives for CS 227X are as follows. At the end of CS 227X the successful student should:

- Be able to quickly and creatively solve small programming problems.
- Be able to convincingly argue the correctness of their solution to a functional programming problem.
- Be excited about some aspects of Computer Science.
- Use good style in writing and modularizing programs.

- Understand the basics of data abstraction, recursion, functional programming, imperative programming, and procedural abstraction.

Students would be able to use reference materials when writing programs.

The main reason for having the 227X course is to try to produce better programmers. Hence our principal objective is that students should be good, at least in programming small problems.

The ability to argue the correctness of a program, convincingly but not formally, is important so that students see that they do not have to always program by trial and error. Debugging is also a useful skill, but one that would not be emphasized in 227X.

It is critical that our majors be interested and excited about computer science; otherwise they will not learn as well or as quickly. We also want to avoid losing our best students to other departments.

Good style is partly in the eyes of the beholder, but that students should have some sense of style can hardly be argued.

The concepts of data abstraction, recursion, imperative programming are fundamental to computer science. Functional programming is used as a teaching tool, to gradually introduce concepts and to allow more formal reasoning early on in the study of programming. It is a potentially important paradigm for future programming as well. Procedural abstraction is important for getting the idea that students should only write code once, and for allowing students to encapsulate and name plans that they have used several times. This is an important pedagogical point.

2.4.2 Enrichment Objectives

Enrichment objectives could be multiplied endlessly. Listed here are those that would be easy to teach based on the texts.

- Be able to use equational reasoning to more formally argue the correctness of functional programs.
- Be able to exploit the correspondence between the recursive structure of data and the structure of a recursive program.
- Understand the basics of object-oriented programming.
- Understand the basics of sorting and searching.

2.5 Syllabus

The material taught would include the following (following the advice of George Springer). The chapter numbers refer to chapters in *Scheme and the Art of Programming*.

- (Chapter 1) Recursive characterization of data (symbolic, numerical, and logical).
- (Chapter 2) Building procedures for processing symbolic data (lists) by "abstracting over data", that is, by using parameters and by making the structure of the computing process correspond in a simple way to the structure of the data. This chapter uses recursion heavily. Some elementary debugging tools are also programmed.
- (Chapter 3) Iterative processes, numbers and operations on numbers, and abstract data types. The major example is a rational number type.

- (Chapter 4) Data driven recursion, including recursion over trees, tail recursion, order-of-magnitude efficiency estimation.
- (Chapter 5) Modularity and structure. Managing names by using local variables and lexical (static) scope. Local abbreviation constructs, polynomials as another example data abstraction, which are used to discuss binary numbers.
- (Chapter 6) Interactive programming techniques, input/output, and the string data type. A square root algorithm, ways of viewing intermediate results and providing data at run-time.
- (Chapter 7) Procedural abstraction. Procedures as arguments and results, Ackermann's function, currying, abstraction of common parts of procedures. Abstraction of flat recursion over lists, abstraction of deep recursion.
- (Chapter 9) Using vectors (i.e., arrays) and mutation to provide random access. Efficiency issues in using and implementing vectors, matrices.
- (Chapter 11) Mutation, side effects, and imperative programming. Efficiency issues and representation of lists.
- (Chapter 12) Object-oriented programming Definition of objects, use of objects to define data structures such as stacks and queues.

Of course, it is possible that not all of the above would be taught in a given semester. In *Scheme and the Art of Programming* text there is also material on language extensions (macros) and streams (for lazy evaluation); or one could end the course with the material from either chapter 10 on sorting and searching or chapter 8 on sets and relations. Some of this material could be used to give the course a broader survey flavor (some data structures, algorithms, and discrete math) instead of a programming focus.

2.6 Discussion

The hope is that students would have a solid foundation for future programming courses, and a glimpse of the rest of Computer Science. Note that this is *not* a course in Scheme, but a course introducing programming.

2.7 Administrative details

We presume we would teach 227X on HP-UX. There would have to be a small segment of the course devoted to teaching the students how to use the Unix system.

For Scheme interpreters, we plan to use one of the publically available interpreters, such as MIT's C-scheme. That particular interpreter is already installed on the department machines and on project Vincent.

3 Details on 228X

3.1 Introduction

Computer Science 228X is a course that introduces students to data structures and object-oriented programming. The study of data structures seeks to answer the following questions.

- How can one correctly implement a specified data structure?
- How can one ensure that an implementation is efficient?
- What is a good way to use data structures in solving problems?

CS 228X addresses all of these questions. It primarily addresses object-oriented answers to the third question.

3.2 Course Description

The catalog description of the course is as follows:

> An object-oriented approach to data structures and algorithms using C++. Object-oriented programming. Program correctness. Other topics include: stacks, queues, trees, searching, sorting, analysis of algorithms, graphs, and file processing. Emphasis on writing and running programs. This course is designed for majors. (4 credits).

Note that this is four credits instead of the original proposal of three credits.

3.3 Prerequisites

The formal prerequisite in the ISU catalog will read "227X and concurrent enrollment in Math 165". This is a change in that Math 165 is no longer a prerequisite, but a co-requisite.

3.4 General Objectives

The general objectives for 228X were developed from our current 212, but reflect an increased emphasis on object-oriented design and programming, especially abstract data types.

3.4.1 Essential Objectives

In general terms the essential objectives for CS 228X are as follows. At the end of CS 228X the successful student should:

- Have a working knowledge of basic algorithms and data structures.
- Be able to judge the efficiency of algorithms.
- Be able to use object-oriented design and programming techniques, including the use of abstract data types, subtyping, and inheritance.
- Be able to carefully (albeit informally) reason about the correctness of programs that use abstract data types and about the correctness of implementations of abstract data types.
- Understanding asymptotic notation and notions of time and space complexity.
- Be able to creatively develop algorithms to solve problems.
- Write, organize and document C++ programs.

Students would be able to use reference materials when writing programs.

The main purpose for the 228X course is to teach students some of the fundamental tools and skills they will need in the rest of the curriculum. These certainly include a core set of basic data structures and algorithms.

Using C++, it is possible to concentrate more on efficiency issues than can be done in 227X (using Scheme). Efficiency considerations are important to computer science, and help distinguish it from mathematics.

Object-oriented programming techniques are helpful in organizing large programs, and in maintaining and enhancing such programs. They are important and marketable skills. Object-oriented techniques also help organize the material in a data structures course, as classes embody abstract data types; without using object-oriented techniques the algorithms and data structures discussed in such a course are disconnected.

We would not emphasize formal verification of programs, but ideas such as planning before coding, loop invariants, and testing to support debugging or correctness arguments.

Asymptotic notation and notions of time and space complexity are fundamental to the study of data structures and algorithms.

Students need not only an arsenal of already developed algorithms, but the ability to make up their own to solve novel problems.

Skills in C++ programming will be important for later in the curriculum and also for jobs. However, note that C++ is *not* the focus of this course; only such C++ as is needed to achieve the other objectives would be taught. On the other hand, this would be a quite useful subset of the language.

3.4.2 Enrichment Objectives

Enrichment objectives could be multiplied endlessly. Listed here are those that various instructors might wish to teach.

- Understanding standard algorithm design strategies.
- Understanding graph algorithms.

3.5 Syllabus

The material taught would include the following. This syllabus is based partly on what is currently taught in 212, partly on potential textbooks for the course, and partly on a consideration of what is planned for 227X.

1. C++. Syntax of C++ expressions, variable and function declarations, and basic statements, including the types int, double, arrays. Procedures for editing and compiling C++ code. Imperative programming in C++. Recursion and iteration. Record types in C++.

2. Data abstraction. Data abstraction using classes in C++. Information hiding.

3. Implementation Data Structures in C++. Pointers, pointer arithmetic, C++ arrays, unions (i.e., variant records).

4. Fundamental Data Structures

   (a) *Linked Lists.* Pointers and dynamic memory allocation. Singly and multiply linked structures.

   (b) *Stacks and Queues.*

(c) *Trees.* Representations and traversals. Tree-based representation of sets.

5. FILES as another implementation data structure.

6. SEARCHING. Sequential and binary search. Binary search trees. Balanced binary search trees.

7. SORTING. Basic sorting algorithms: insertion and selection sort: Faster sorting algorithms: quicksort, heaps and heapsort, mergesort.

8. INTRODUCTION TO ANALYSIS OF ALGORITHMS. Asymptotic notation, time and space complexity.

If time permits, we would also discuss:

- ALGORITHM DESIGN TECHNIQUES. Backtracking and branch-and bound. Using induction to design algorithms. Divide-and-conquer and dynamic programming.

- GRAPHS. Representation, traversals, minimum-cost spanning trees, and shortest path problems.

There is some possibility of using Scheme to illustrate some points about Sorting and Searching that would otherwise be difficult in C++.

3.6 Discussion

This is not a course in C++ programming. Students would not be introduced to *all* the features of C++.

3.7 Administrative details

We presume we would teach 228X on HP-UX. There would have to be a small segment of the course devoted to teaching the students how to use the Unix system.

Acknowledgements

The courses described here were developed with the help of professors Gadia and Oldehoeft. Many other faculty in the department have contributed ideas and discussions.

Course Specifications for New Introductory Courses: Computer Science 227X and 228X

TR 92-09
Albert L. Baker, David Fernandez-Baca, and Gary T. Leavens

April, 1992

Iowa State University of Science and Technology
Department of Computer Science
226 Atanasoff
Ames, IA 50011

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

Tech Report: TR 92-09
Submission Date: April, 1992