A Brief History of the Computing Curriculum
at Grinnell College

Notes by Henry M. Walker

Grinnell College first included a computing or computer science course in its curriculum in 1971, when a 1-credit FORTRAN course was offered with Calculus II as a co-requisite. Since that time, there has been a continuing evolution of the computing curriculum. Several significant influences on this development have been the increased use of computing throughout the college: the expansion of available computing equipment, the changing nature of the needs of students, and the maturing of the discipline of computer science itself.

The history of the computing and computer science curricula may be organized into the following periods:

- Introductory Programming In Context: 1971-1978
- Computer Studies: 1978-1990
- Establishment of the Computer Science Major: 1990-1995
- Redefinition of Computer Science: 1995-1997
- Further Increased Enrollments: 1997-2001
- Relative Stability 2001-2004
- Emergence of a Separate Department: 2006-Present

As with most historical developments, the motivations, ideas, and experiences at each of these stages strongly influenced the next.

This review highlights some major themes and directions that have led to the current computer science program at Grinnell College. With the number of courses and people involved over a period of almost four decades, any thorough description of these periods could extend for many more pages and is beyond the scope of this article.

1. Introductory Programming in Context: 1971-78

The first computing course at the college emphasized programming and applications and appeared in 1971, the same year as the institution of the Freshman Tutorial. Specifically, Math 134 was offered as a 1-credit supplement to Math 133.

Math 134, Computing Programming Workshop
(1 credit)
An introduction to computer programming, with special emphasis on material related to calculus. Prerequisite: concurrent registration in Math 133 (Calculus II).

This FORTRAN-based course emphasized numerical methods and utilized a remote card-reader and printer connected to a computer at the University of Iowa.

In 1974, a more extensive introductory course was offered for a general audience.

Math 100, Introduction to Computer Science
(4 credits)
An introduction, first, to computers and their uses, and second, to a programming language, either BASIC or FORTRAN. Includes a study of algorithms and data structures. Instruction in writing programs and using "prepackaged" programs to solve numerical and non-numerical problems. Prerequisite: none.

Both Math 100 and 134 stressed programming to solve problems, with the Math 100 open to a wider audience and addressing a somewhat broader range of applications.
During this time, Math 134 served as the primary programming course for students taking mathematics courses. In 1975, the audience for Math 134 was broadened considerably, when the co-requisite was expanded to allow Math 131 (Calculus I), Math 128 (Calculus and Probability II), or Math 133 (Calculus II).

In 1976, Math 134 was replaced by Math 125, which allowed a still broader range of mathematics courses to serve as appropriate co-requisites.

Math 125, Computing Programming Workshop (1 credit)
An introduction to BASIC programming, with examples drawn from the mathematics course in which the student is concurrently enrolled. Prerequisite: concurrent registration in any mathematics course numbered from 121 through 234 (precalculus through differential equations or mathematical statistics).

Lectures in this course focused on language syntax, examples, and a few standard algorithms (e.g., searching and sorting), while problem sets were tied to specific mathematics courses. The shift to BASIC reflected the College’s acquisition of its first computer, a PDP 11/45.

By 1977, the faculty realized a need to clarify the audience, goals, and scope of Math 100 and Math 125. Both courses covered many similar elements of programming. Applications in Math 125 had expanded to many areas of mathematics, well beyond algorithms specifically related to calculus. Since most Grinnell students took mathematics during their undergraduate program, the difference in target audience between Math 100 and 125 was much less clear than several years earlier.

With no prerequisite, Math 100 proceeded at a slower pace than Math 125, and Math 100 discussed some software packages. Thus, Math 100 was offered for 4 credits, while Math 125 was offered for only 1 credit. In practice, many students selected the course on the basis of credits offered rather than educational goals or interests.

To clarify course content and to highlight different perspectives in computing, the computing curriculum was redesigned in 1977 to separate programming and social issues into two courses. First, the prerequisite for introductory programming was dropped completely in 1977, when Math 125 was replaced by Math 101.

Math 101, Introduction to Computer Programming (1 or 2 credits)
An introduction to BASIC programming. Problem sets are tailored to individual students’ interests and background, with problem sets associated to courses in several departments being available. This course may be repeated once for 1 credit if taken for 1 credit initially. Prerequisite: none.

At the same time, a course on the social issues of computing was introduced to offer a more general perspective.

Math 102, Computers in Society (2 credits)
A systematic study of the impact of computers on society using assigned readings and lectures. Discussion topics will be chosen from: individual privacy and technology, human obsolescence, artificial intelligence, effects on education, and computer control of systems such as monetary flow and national defense. Prerequisite or co-requisite: Math 101.

Together, Math 101 and 102 covered the content of Math 100, while covering mathematical algorithms and techniques for students with appropriate technical background. Math 100 and 125, therefore, were dropped.

As the appeal of programming increased, Math 101 enrollments soared, as shown in the following table which gives the number of students receiving credit for the course:
Number of Students Completing Course

<table>
<thead>
<tr>
<th>Year</th>
<th>Fall</th>
<th>Spring</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1977-78</td>
<td>43</td>
<td>55</td>
<td>98</td>
</tr>
<tr>
<td>1978-79</td>
<td>52</td>
<td>43</td>
<td>95</td>
</tr>
<tr>
<td>1979-80</td>
<td>57</td>
<td>125</td>
<td>182</td>
</tr>
<tr>
<td>1980-81</td>
<td>72</td>
<td>80</td>
<td>152</td>
</tr>
<tr>
<td>1981-82</td>
<td>66</td>
<td>127</td>
<td>193</td>
</tr>
</tbody>
</table>

With this interest, Math 101 was one of the few courses on campus in which the enrollment for a semester exceeded the course number.

**Computer Studies: 1978-1990**

By 1978, interest on campus in computing had increased in several ways. Faculty in several departments used the computer to store and analyze data in various ways. Courses in such fields as economics, psychology, and sociology, introduced students to techniques involving the computer. Some students had an increasing interest in computer languages, algorithms, and other aspects of computer science. In addition, Computer Services actively promoted the increased use of computing across many areas of the college. Such factors motivated interest in coordinating the use of computing within the curriculum as well as a modest expansion in the study of specific topics within computer science.

As a parallel development, during the previous year, the College had instituted a program of Interdisciplinary Concentrations to encourage students to engage in a breadth of study, following a coherent schedule of courses. Thus, the 1977-1978 College Catalog states:

> Each recognized concentration includes an organized cluster of courses drawn from several disciplines and related to a common focus of interest. Thus each provides a structured introduction to a broad area of study while including sufficient flexibility to adapt each program to a student’s particular focus of interest. Each culminates in an interdisciplinary senior seminar in which students and faculty draw upon their work in the several disciplines. [p. 140]

Combining these themes and interests, an Interdisciplinary Concentration in Computer Students was established in 1978, with this description in the 1978-1979 College Catalog:

> This program introduces the student to some of the concepts fundamental to computer science and to some of the major areas of computer applications. The computer serves as a common thread for the required courses in computer science, mathematics, physics, and at least one application area. The senior seminar draws on ideas generated in those fields to study an important topic in computer science. [p. 103]

In addition to taking advantage of the existing computing courses and an electronics course, offered by the Physics Department, the following computer science course was introduced:

> Math 201, *Programming Language Concepts and Data Structures* (4 credits)

An introduction to the design and manipulation of data structures, such as tags, linked lists, stacks, and hash tables. A study of the structure and characteristics of programming languages through a comparison of BASIC with one of more other languages, such as FORTRAN, SNOBOL, PASCAL, or Assembly Language. Prerequisite: Math 133 or 128 and two credits of Math 101.

In retrospect, Math 201 began a trend that would be repeated frequently, in which a single course was severely overloaded in an attempt to provide students with some understanding of numerous topics from computer science within the constraint of a single semester. With some luck and much determination, an instructor could select and restrict topics adequately to create a coherent, manageable course.
The new Computer Science Concentration also established a precedent by assuming Mathematics faculty would staff the Senior Seminar as an overload as necessary. While there was some hope that other staffing might be found occasionally, in practice the Mathematics faculty had to teach overloads to make needed computer science courses available and to meet requirements for the concentration.

Viewed from a historical perspective, Math 201 was successful in providing some expansion of the computing curriculum and in introducing students to some important topics. With the wealth of material appropriate for that course, however, it was only a matter of time before Math 201 was identified as being hopelessly overloaded.

For the most part, the Computer Studies Concentration accomplished its intended purposes of allowing a partial expansion of the computing curriculum and encouraging the coordination of computing courses at the College. In addition, external funding for interdisciplinary concentrations allowed a significant expansion of the library holdings in computing. For the next several years, small adjustments were made in the counting of specific applications courses within the Computer Studies Concentration.

The main structure of the computer science courses remained unchanged until 1983. During that year, the Mathematics Department redesigned its lowest-level courses to emphasize problem-solving. Specifically, the precalculus course was changed from Algebra and Trigonometry to Problem Solving and Precalculus. Similarly, Math 103 was introduced to address difficulties that some beginning programming students had with problem solving.

Math 103, Problem Solving and Computing  (4 credits)
An introduction to the nature of problem solving. Topics will include readings about problem-solving techniques, an introduction to a programming language, and—most importantly—analyzing and solving problems. The problem-solving techniques will be of a general nature and will apply to problems in a wide variety of disciplines. Both the power and the limitations of the computer as a problem-solving tool will be discussed. Prerequisite: None.

At the same time, research in many areas throughout the curriculum had evolved to include computing applications. This reduced the need for a separate applications-oriented course, and Math 102 was dropped as no longer necessary.

Since many students now were entering college with some computing background, it seemed these students could start at a level somewhat higher than Math 101. Thus, in 1983, the Math 101 course was reduced to 1-credit only, and a more substantial course became the natural starting place for students interested in computing or the Computer Studies Concentration.

Math 203, Introduction to Computer Science  (4 credits)
An introduction to fundamental computer-science topics. Includes programming in Pascal, machine organization, elementary data structures, and some classical algorithms for sorting and searching data. Prerequisites: Math 101 or 103 (or equivalent); a course in quantitative problem-solving [e.g., math, physics, or chemistry]; sophomore standing.

To further expand the computer science content of the Computer Studies Concentration, a second course was added.

Math 204, Algorithms and Software Design  (4 credits)
Study of data structures and algorithms with an emphasis on structures that can be applied. Consideration of large application programs with attention to program development and human interface. Prerequisite: Math 203.

With these additions, the computer science requirement for the Computer Studies Concentration now specified Math 203, Math 204, and Physics 220 (Electronics).

The discussions that led to refining the Computer Studies Concentration in 1983 also prompted the Mathematics Department to substantially expand its upper level offerings in computer science. The specific proposal was introduced by the following rationale:

In recent years, Computer Science has similarly emerged as a rigorous, intellectually stimulating discipline with a firm foundation in theory and methodology. Computer Science education need not be a vocationally oriented, “how to” series of courses; instead, Computer Science can stress concepts, theory, and abstractions which traditionally have been important in a liberal arts curriculum. Therefore the Mathematics Department believes that now is an appropriate time to continue broadening its offerings to include topics in Computer Science in the [mathematics] major. ... [February 11, 1983 proposal, pp. 8-9]

Three courses were introduced at this time:

Math 301, Data Structures (4 credits)
Study of structures used to organize data (lists, stacks, queues, trees, graphs) and of the algorithms used to manipulate these structures. Assignments to implement data structures and to use them in computer science and other applications programs. Emphasis is on mathematical principles behind the data structures. Prerequisites: Math 203 and 218 (Combinatorics).

Math 302, Programming Language Concepts (4 credits)
A careful study of the concepts underlying programming languages. Examples from Pascal and other high-level languages will be used to illustrate the general concepts. Prerequisite: Math 301.

Math 341, Automata, Formal Languages, and Computational Complexity (4 credits)
A formal study of computation devices, their related languages, and the possibility and difficulty of computations. Examples are pushdown automata and Turing machines, context-free languages and recursively enumerable sets, and the halting problem and NP-completeness. Prerequisites: Math 203 and 215 (Linear Algebra and Differential Equations).

With concurrent changes in the Mathematics Major, these courses could be counted as part of an [undesignated] track through that major. Due to the interdisciplinary nature of the Computer Studies Concentration, however, these upper-level courses were not accepted for that program.

In addition to adding significant computer-science content to the curriculum, these additions also reflected various national discussions of curriculum that were taking place during the mid-1980s. In particular, Math 301-302 reflected many elements of courses CS7-8 in ACM’s Curriculum ’78, and all three courses were strongly influenced by informal discussions with a group of faculty that would become the Liberal Arts Computer Science Consortium (LACS).

By 1986, several problems had surfaced regarding Math 203.

The principal weakness of the current Mathematics 203 course is that it tries to cover far too much ground. The universal experience at other colleges and universities is that making students really proficient in Pascal requires a semester (and in many places Pascal programming is a two-semester sequence); we try to do it in one and simultaneously to develop as much as possible of the mathematical background presupposed in more advanced courses in computer science. The result is that neither job is done as well as it might be. [Departmental Memo to the Science Division, March 5, 1986, p. 2]
The proposed solution was to redistribute the subject matter of Math 203 into two courses:

Math 151, *Computer Programming in Pascal* (4 credits)
An introduction to computer programming using the high-level, block-structured programming language, Pascal. Topics include algorithm design, coding, testing, debugging, producing documentation, procedural abstraction, and specific examples of data structures. Prerequisite: completion of a Level I course in Problem Solving and Technology Studies [e.g., math, physics, chemistry].

Math 206, *Fundamentals of Computer Science* (4 credits)
An introduction to many of the fundamentals concepts in computer science. Builds upon the programming knowledge from Mathematics 151 to study the design, analysis, and verification of algorithms. Includes a discussion of data abstraction and data structures. Also provides an overview of the field of computer science. Prerequisite: Math 151.

While this change in introductory computer science retained Pascal as the first programming language (in Math 151), the 1986 proposal for Math 206 included a course syllabus based on Modula 2.

With this change in Math 203, the computer science requirement in the Computer Studies Concentration also was modified, so that Math 151 and 204 were required, and students could choose from either Math 206 or Physics 220 *Electronics*. In addition, this change shifted Math 301 to “a slightly higher level, more in keeping with its prerequisite chain. The emphasis of the course also would broaden beyond data structures somewhat to include an expanded coverage of the analysis of algorithms.” [Ibid.]

This redesign of the introductory sequence and Math 301 was consistent with many national curricular discussions, and the Rationale of the proposal quoted heavily from a draft “Model Curriculum” by LACS. The Rationale also reiterated the role of Math 103 for preparing students with relatively weak backgrounds for the regular programming course (originally Math 203, now Math 151).

**Emergence of the Computer Science Major: 1990-1995**

During the 1989-1990 academic year, the Mathematics Department had extensive discussions concerning the position of computer science within the department and the needs of computer science students. Some basic conclusions are outlined in a March 15, 1990, memorandum from the department to the Dean:

... Computer Science has emerged over the past few years as a separate discipline within the liberal arts and at the College. While both Mathematics and Computer Science may be viewed as part of a broad area sometimes called the Mathematical Sciences, the two fields focus on rather different types of problems, and the two disciplines approach problems in distinct ways.

Further, in talking to people outside the College (including prospective students, potential employers, and publishers), there currently is some confusion about both the mathematics and the computer science curriculum at Grinnell. For example, the course listings for the two disciplines are intermixed. Prospective students interested in mathematics sometimes are put off by the integration of computing courses throughout the mathematics listings; those interested in computer science find it difficult to locate relevant computing courses. The solution to this identification problem is to list mathematics courses separately from those in computer science, all within the same overall departmental listing. [p. 3]

More specifically, these discussions led to four significant changes. First, one course was added to the curriculum.

Study of a simple computer’s architecture and organization and the principal components of a typical operating system. Topics from architecture: levels of organization, digital logic, instruction execution, information storage, addressing, machine and assembly language. From operating systems: storage management, scheduling, concurrent processing, synchronization, and data protection. Prerequisite: Math/CS 206.

With this new course, the computer science curriculum now covered the full range of core undergraduate computer science courses, as recommended by LACS.

Second, *Algorithms and Software Design* was upgraded, by changing its prerequisite from Math/CS 151 to Math/CS 206, and the course number was increased correspondingly from 204 to Math/CS 223. This project-based course became the only non-core course in computer science open to majors, a circumstance that continued for several years.

The addition of Math/CS 207 and the upgrading of Math/CS 223 provided a thin, but adequate, base for a designated major in computer science. While this major contained somewhat less computer science than might be desirable, it also required four semesters of mathematics, including Calculus I and II, Linear Algebra and Differential Equations, and Combinatorics. This mathematical requirement allowed some upper level courses, such as Math/CS 301 and 302, to be offered at a more rigorous level than was possible at many other schools. As an aside, it seems noteworthy to mention that this Computer Science Major was the first new major to be approved at Grinnell College in approximately 25 years.

Third, the name of the department was changed to the Department of Mathematics and Computer Science to further clarify the position of computer science within the college. Similarly, College Catalog materials were updated to reflect the separate nature of the two disciplines, with mathematics courses being listed separately from those in computer science.

Fourth, since the term “Computer Studies” was often confused with “Computer Science”, since national expectations for computing expertise had risen considerably in recent years, and since computing applications had been integrated into the curriculum in many ways, the College dropped the Computer Studies Concentration.

Following the adoption of the Computer Science Major, work focused upon developing and refining existing courses, and few formal changes were proposed for several years. There were, however, several important experiments in a few courses, most notably in CS 206. There, instructors tried a variety of languages, including Modula 2, Scheme, and Pascal with Units (e.g., Sun Pascal or HP Pascal). Also, a Plus-2 option based on C was sometimes offered in conjunction with CS 206.

**Redefinition of Computer Science: 1995-1997**

By 1995-1996, the faculty teaching computer science courses had gained considerable experience with the new major. New perspectives and insights came from observing student difficulties, reading published reports, and talking to colleagues at national meetings.

At this time, staffing for computer science totaled 1.8 FTE per year, yielding the following approximate two-year schedule with 9 sections of courses each year:
This schedule covered basic topics. Discussions on improvements and refinements concentrated in three areas:

- College requirements on majors
- Staffing constraints
- The nature of the introductory sequence

**College Requirements on Majors**

The most pressing difficulty centered upon a technical restriction at Grinnell College that no more than 48 credits of work in a single department could be counted toward graduation. While this limitation was designed to force students to take a range of courses as part of their liberal arts education, the rule did not anticipate the possibility that two disciplines might be housed in the same department. Thus, the following proposal passed the Science Division and the Curriculum Committee in 1994-1995. The September 12, 1995, memo to the full faculty presented the technical motion and clarified the operative requirements for the computer science major:

Resolved, that, in applying the College’s limit of 48 credits within one department that students may count toward graduation, up to 12 credits of mathematics (including AP credits in mathematics) be exempted for students majoring in computer science. Double majors in the two disciplines would not be allowed.

Although mathematics and computer science are housed in the same department, they are two distinct disciplines. The motivation for this proposal is the requirement that computer science majors take 4 mathematics courses (16 credits) in addition to the seven computer science courses (28 credits), which brings the total to 44 required credits. The seven courses constitute a minimal major that most students need to supplement with independents, plus 2’s, or summer research, but doing so puts them against the 48 credit ceiling.

The September 12 proposal also noted that the computer science major required all seven computer science courses offered [except CS 103] plus mathematics throughout MAT 218 (which requires 3 preceding mathematics courses.) The computer science major met guidelines by LACS and others, but a student had no flexibility in choosing courses.
Staffing Constraints

From the creation of the Computer Science Major, enrollments and majors increased steadily. Most core courses could be offered in alternate years, and scheduling constraints made it difficult for some students to take all required courses for the CS major. These factors contributed to a September 18, 1996 proposal from the Department of Mathematics and Computer Science. Key features of this proposal included:

- An increase in staffing in computer science from 1.8 FTE per year to 2.8 FTE per year.
- Splitting CS 207 into two courses to more adequately cover core material recommended in national curricular guidelines.

CS 211, *Computer Organization and Architecture* (4 credits)
Study of both traditional and alternative computer architectures. Introduction to digital logic, microcode, Von Neumann architectures, data representations, fetch/execute model, RISC/CISC, instruction formats and addressing, machine and assembly language, memory architecture and algorithms, I/O architecture, and elements of distributed systems.

CS 213, *Operating Systems and Parallel Algorithms* (4 credits)
Study of the principal components of typical operating systems and an introduction to parallel algorithms. Topics from operating systems: storage management, scheduling, concurrent processing, synchronization, data protection, and security. Discussion of models of parallelism and algorithms for problems in such areas as lists, trees, searching, sorting, graphs, geometry, and strings.

- A modest increase in the offering of courses. The September 18 proposal included this preliminary schedule:

<table>
<thead>
<tr>
<th>Year 1</th>
<th></th>
<th>Year 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall</td>
<td>Spring</td>
<td>Fall</td>
</tr>
<tr>
<td>Tutorial</td>
<td>103</td>
<td>Tutorial</td>
</tr>
<tr>
<td>151</td>
<td>151</td>
<td>151</td>
</tr>
<tr>
<td>206</td>
<td>206</td>
<td>206</td>
</tr>
<tr>
<td>211</td>
<td>243</td>
<td>213</td>
</tr>
<tr>
<td>301</td>
<td>302</td>
<td>301</td>
</tr>
<tr>
<td>Elective</td>
<td>Elective</td>
<td>Elective</td>
</tr>
<tr>
<td>323</td>
<td>323</td>
<td>323</td>
</tr>
</tbody>
</table>

This schedule uses the following course numbering for existing or new courses:

103: Problem Solving and Computing (or other general literacy course)
151: Fundamentals of Computer Science I
206: Fundamentals of Computer Science II (now offered every semester)
211: Computer Organization and Architecture (from old 207)
   (As in the “Revised Model Curriculum” – with memory allocation and alg.)
213: Operating Systems and Parallel Algorithms (from old 207)
243: Program Verification and Analysis of Algorithms (new, alternating with 341)
301: Algorithms and Abstract Data Types (offered every year)
302: Programming Language Concepts (offered every year)
341: Theory of Computation (still offered in alternate years)
323: Algorithms and Software Development (upgraded from current 223)
The nature of the introductory sequence

Through the fall of 1996, the introductory sequence followed a traditional approach: CS 151-206 focused on an imperative approach to problem solving supported by the Pascal programming language. This sequence, consistent with the Advanced Placement Computer Science (AP CS) curriculum, was widely adopted by many schools in the middle 1980s and the early 1990s, and it provided students with reasonable depth in one view of problem solving. However, the field of computer science had identified several fundamental approaches to problem solving, and this traditional curriculum introduced multiple views of problem solving only in a junior-level programming course (e.g., CS 302 at Grinnell).

After extensive discussion, the CS faculty decided to follow a different model for the introductory curriculum. In this curriculum, the first two semesters (e.g., CS 151 and CS 206) would introduce students to two different problem-solving paradigms, as recommended in several natural forums of the time, including the “Revised Model Curriculum” of the Liberal Arts Computer Science Consortium.

In discussions, the faculty agreed that the first course would focus on functional problem solving using the Scheme programming language. The leading alternatives for the second course, however, had both advantages and disadvantages. If the course focused on object-oriented problem solving with C++, the curriculum would connect with contemporary directions for software development, but the C++ language was large, complex, and awkward for beginners. If the course emphasized imperative problem solving with the C programming language, then students would learn a conceptually elegant computational model, but the language would not support object-oriented problem solving.

The emergence of the Java programming language in the mid 1990s resolved several misgivings and concerns. Java supports object-oriented problem solving, has a relatively clean syntax, and provided a reasonable basis for the second course.

Course descriptions for the new introductory sequence clarified the multi-paradigm approach:

CSC 151, Fundamentals of Computer Science I (4 credits)
A lab-based introduction to basic ideas of computer science, including recursion, abstraction, state, information hiding, and the design and analysis of algorithms. Includes introductory programming in a high-level, functional language.

CS 206, Computer Organization and Architecture (4 credits)
Builds upon Computer Science 151 to study object-oriented problem-solving, the design and analysis of common algorithms, fundamental abstract data types and data structures, and elements of testing and verification. Also provides an overview of the field of computer science. Includes team projects and formal laboratory work.

With the anticipated expansion of the CS faculty in 1997-1998, this new curriculum was implemented beginning with new sections of CS 151 in Spring 1997; the new CS 206 was first offered in Fall 1997 by the newly-hired computer scientist.

Further Increased Enrollments: 1997-2001

The November 30 report included three key observations:

- At the introductory level, enrollments nearly doubled between 1997 and 1999. The number of students taking CSC 151 went from about 30 in each 2 sections for 1997-1998 to over 30 in each of three sections in 1998-1999, and numbers continued strong in fall 1999. Also, since this course was lab-based and made extensive use of collaborative learning, an undergraduate assistant regularly helped the instructor address student questions in these large-enrollment sections.

- The number of computer science majors similarly had approximately doubled, and numbers for subsequent years seemed strong.

- “Overall enrollments were approximately constant for the four years between 1993-1994 and 1996-1997. Then, enrollments increased about 30% in 1997-1998. In 1998-1999, enrollments increased another 20%, even though faculty strength shrank from 2.6 FTE to 2.0 through leaves.” Enrollments in fall 1999 were at least as strong as in the previous year. [November 30, 1999, departmental report to the Dean]

These enrollments prompted the department to request an increase in tenure-track staffing in 1999. With compelling enrollment pressures and strong programmatic needs, the Executive Council and Administration authorized a new tenure-track line in computer science in 1999-2000, and the department conducted a successful search in 2000-2001.

Two refinements of the curricular offerings completed the proposal:

- A new course, CSC 153, Computer Science Fundamentals, target to incoming students who had strong high school computer-science background.

- Common elements from several 200- and 300-level courses were identified and integrated into a new course, originally proposed as CSC 156, but later formalized as CSC 201, Data Representation, Memory Management, and Program Verification:

CSC 156, Data Representations and Structures (4 credits)
Study of machine-level representations of data and techniques for managing storage, using formal methods of program design and a low- or mid-level programming language, such as C. Topics include Boolean logic and proof, language semantics, assertions and invariants, numerical approximations and errors, pointers, memory allocation and deallocation, and the run-time stack.

The 1999 proposal included the following expanded plan for courses and staffing in computer science:
This schedule uses the following course numbering for existing or new courses:

- **105**: Overview of Computer Science (or other general literacy course)
- **151**: Fundamentals of Computer Science I
- **152**: Fundamentals of Computer Science II
- **153**: Computer Science Fundamentals
- **156**: Data Representations and Structures
- **205**: Computational Linguistics (offered in alternate years)
- **211**: Computer Architecture and Organization (offered in alternate years)
- **213**: Operating Systems and Parallel Algorithms (offered in alternate years)
- **223**: Software Design (offered in alternate years)
- **261**: Artificial Intelligence (offered in alternate years)
- **301**: Algorithms
- **302**: Programming Language Concepts
- **341**: Theory of Computation
- **362**: Compilers (offered in alternate years)
- **364**: Computer Networks (offered in alternate years)

Such staffing requires 18 sections of computer science in each year (11 at the 100-level, 7 at the 200- or 300-level) and fully utilizes an expansion to 3.6 FTE.

**Relative Stability: 2001-2004**

After the expansions of 1996 and 1999 and the related programmatic revisions, the computer science faculty focused on developing and refining its program. With several faculty leaves scheduled during this period, the department hired a three-year visitor to cover courses.

During this period nationally, difficulties of the dot-com industry and changes in the public perception of computing led to a general reduction in prospective students interested in computer science throughout the country. At Grinnell, enrollments followed a similar trend, although specific causes for the downturn remain a matter of speculation.

**Moving toward Independence: 2004-2006**

Through the 1990s and into the 2000s, the disciplines of mathematics and computer science diverged, making it impractical for faculty to remain current in both fields. As a result, the department met together for meetings to consider matters of joint interest, but many of these discussions focused upon mathematics. In addition, the computer science faculty met separately to consider curricula, course changes, course staffing, teaching assignments, etc. By 2005, the two sides of the combined department were largely operating independently.
In Fall, 2005, both parts of the combined department collaborated on the following proposal:

The divergence of the disciplines of mathematics and computer science motivates the following proposal for restructuring the Department of Mathematics and Computer Science to form two separate departments. After the restructuring, the programs and faculties of the two departments would be subject to the same regulations as all other departments.

**Moved:**

A. That the current faculty and programs within the Department of Mathematics and Computer Science be restructured into two independent departments, a Department of Mathematics and Statistics and a Department of Computer Science, both within the Division of Science;

B. That the requirements for a major in computer science be those currently listed in the College catalog under the heading “The Computer Science Major,” except that the third paragraph (exempting credits in mathematics from the limit of forty-eight credits in one department and prohibiting double majors in computer science and mathematics) be deleted; and

C. That the requirements for a major in mathematics be those currently listed in the College catalog under the heading “The Mathematics Major,” except that the 32 credits be in the “Department” and that 4 credits of computer science might count with departmental approval.

The primary motivation for this proposal comes from a divergence of computer science and mathematics. The main argument for restructuring proceeds in three steps.

1. The disciplines of computer science and mathematics have diverged, so that computer science can no longer be considered a branch of mathematics.

2. The computer science and mathematics programs and faculty at Grinnell College now function as largely separate and independent entities.

3. The public view of these disciplines within the same department increasingly creates awkwardness and confusion — both within the College and externally. Restructuring the department into two parts would resolve current difficulties and create higher visibility for each discipline.

Altogether, this proposal addresses current difficulties and reflects both the current intellectual separation of these disciplines and the practical independence of the current programs.

This proposal passed the Science Division in Fall 2005 and the full faculty and Board of Trustees in Spring 2006. The two new departments were established effective July 1 2006.

**Emergence of a Separate Department: 2006-Present**

Computer Science first appeared as an independent department in the Grinnell College Catalog for 2006-2007. Preparatory activities included rewriting such materials as write-ups in the catalog, the Student Handbook, and Admission handouts. During 2006-2008, the department focused on four major projects:
A proposal for expansion through the College’s Expanding Knowledge Initiative (EKI)

A move to new facilities on the third floor of the expanded Noyce Science Center

A careful review of the curriculum

An external review, requested by the CS faculty, to help the CS program to maximize
its utilization of the new facilities and expanded staffing

A proposal for expansion through the College’s Expanding Knowledge Initiative (EKI)

In 2005, the College started an Expanding Knowledge Initiative (EKI) to make strategic expansions to the faculty and to aid the interdisciplinary programs of the college. This interdisciplinary perspective fit well with the perspectives of the CS faculty. For example, Two CS faculty had received course releases through the college’s HHMI grant for attending courses in other departments and bringing ideas from those classes into their own teaching. One faculty member expects to teach introductory statistics; another LIN 114, General Linguistics; and two regularly teach TEC 154, The Evolution of Technology. Each CS faculty member also has additional interdisciplinary courses they would like to develop for the general college community, should staffing permit.

Since these interests resonated well with the EKI, the CS faculty developed a proposal to combine these interdisciplinary theme with two additional factors:

- Leave proofing the department (an additional position would allow the CS department to cover most leaves without the need to hire a visitor)
- Offering two first-year tutorials on a regular basis, instead of one.

The CS faculty proposed teaching some courses outside the department (e.g., tutorial, Introduction to Statistics, The Evolution of Technology, Introduction to General Linguistics), and this could free up faculty in other departments, and the CS faculty also could expand interdisciplinary offerings at the college.

With these features, the Executive Council and administration approved the proposed expansion, and the department was successful in hiring a new tenure-track faculty member for the 2008-2009 academic year. With this position, the department has a full strength of 4.6 FTE per year.

A move to new facilities on the third floor of the expanded Noyce Science Center

In 2006-2007, the facilities for computer science in the 1986 wing of the building had evolved from the move of the Department of Mathematics two decades earlier. Thus, during its first year of existence in 2006-2007, the new Department of Computer Science had these facilities:

- 3 dedicated faculty offices (one created from a previous student work room)
- 1 temporary faculty office (a mathematician was on leave)
- 2 shared teaching computer labs
- 2 shared classrooms
- 1 shared lounge and kitchen/office area

Although these facilities addressed some basic needs, the department had no dedicated classrooms or labs, so equipment could not be configured for the needs of systems courses (e.g., networking), and there was no lab where equipment could be left unhindered from one class meeting to the next. Further, although the combined CS/Math network (MathLAN) had about 100 Linux-based computers, these were all in offices, classrooms, or heavily-scheduled labs. There were no department-based computer labs in which students could work on homework during the day, and faculty had no individual research labs.
As the college planned for an expansion and renovation of the Noyce Science Center, the CS faculty worked with architects and the Building Committee to create new space that would resolve the numerous constraints in the inherited space. Thus, when the new CS Department could move into its new facilities in May/June 2007, the new facilities provided much greater support of the CS program:

- 5 dedicated faculty offices (with the potential to expand to an additional office)
- 5 dedicated faculty research labs
- 1 dedicated teaching lab for introductory courses
- 1 open lab to allow students to work on assignments and projects throughout the day
- 1 “systems” lab that allows computers to connect to the campus network or just within the room
- 1 “robotics/capstone” lab with space to lay out equipment from one class to the next
- 2 student study areas
- 1 CS Commons to support gatherings and community activities

The new facilities allow the CS faculty to greatly expand its program and to renew its efforts to employ current pedagogy in its courses.

A careful review of the curriculum

In summer 2007, the CS faculty held a week-long workshop to review the curriculum. Although the faculty wanted to be informed by national recommendations from the Liberal Arts Computer Science Consortium and by Computing Curricula 2001 from the Association for Computing Machinery and the Institute of Electrical and Electronics Engineers, the faculty wanted to tailor details to fit well within the environment of Grinnell College.

At the introductory level, the multi-paradigm approach resonates well with Grinnell’s interest in multiple views of problem solving. Thus, discussions focused on reorganizing topics to ease transitions while maintaining the general approach and philosophy. The department’s review document describes several proposed revisions to the introductory sequence:

In the new introductory sequence, students would still begin with a course in functional problem solving in Scheme (CSC 151). The second course (CSC 161) will focus on imperative problem solving in C, with a focus on pointers, memory management, and data structures. The third course (CSC 207) will focus on object-oriented design in Java and begin to look at algorithm analysis in greater depth. Relative to the current curriculum, this would reverse the order of the object-oriented and imperative paradigms. We would also greatly reduce the coverage of formal program verification methods from the imperative programming course and go into greater depth on object-oriented design.

These changes respond to several concerns.

1. The current CSC 152 must cover both the imperative and object-oriented paradigms. By reversing the order of the courses, students learn only one paradigm at a time.

2. There is an argument that having implemented data structures in C will better equip students to appreciate the strengths of the object-oriented paradigm, as well as Java’s standard library implementations of those data structures.

3. The prerequisite for CSC 223 (Software Design) is currently CSC 152 or 153, which many students complete during their first year. Thus, CSC 223 is taken by many students in their second year, as well as juniors and seniors. This difference in maturity makes it difficult to pitch the course at an appropriate
level, and especially difficult to form effective project teams. Placing the object-oriented paradigm third in the introductory sequence ensures that the development methodology course is taken mainly by juniors and seniors.

4. Students report that formal program verification (with the Gries textbook) has been a particularly difficult and unpopular section of CSC 201. We reluctantly acknowledge that students at this level may be better served by focusing on the most basic aspects of this topic: preconditions, postconditions, and loop invariants.

At the upper level, national curricular guidelines recommend a large core of topics. Since all majors at Grinnell are limited to only 32 credits, most courses within the CS major have been specified. The external review report suggests the following possible major requirements to provide somewhat more flexibility while still covering many core topics.

- CSC 151 Functional Problem Solving [new title]
- CSC 161 Imperative Problem Solving and Data Structures [replaces 201]
- CSC 207 Algorithms and Object-Oriented Design [replaces 152]
- An under the hood course (CSC 211 Architecture or CSC 213 Operating Systems) [unchanged]
- A course that covers development methodology and includes a large team project (CSC 223 Software Design or CSC 225 Databases and Web Applications Design)
- An upper-level algorithms course (CSC 301, Analysis of Algorithms) [new title]
- An upper-level programming languages course (CSC 302 Programming Language Concepts or CSC 362 Compilers)
- An upper-level theory course (CSC 341 Automata, Computational Complexity, and Formal Languages) [unchanged]

... Because it is a prerequisite for CSC 301 and CSC 341, computer science majors will still be required to take MAT 218 Combinatorics and its prerequisites, currently two semesters of calculus (including multi-variable calculus) and one semester of linear algebra.

The 2007-2008 External Review

Between 2005 and 2007, computer science emerged as a new department, the department moved to new facilities, Grinnell’s Executive Council approved an expansion in the CS faculty, and the department could anticipate several coming years of junior and sabbatical leaves. These circumstances prompted the CS faculty to request an external review as an opportunity to review current circumstances and to develop a clearer vision for the future.