

Proposed revisions in the Computer Science curriculum

- Rearrange the introductory courses
- Allow more options in the major requirements
- Adjust some of the prerequisites
- Add electives

- Rearrange the introductory courses
- Allow more options in the major requirements
- Adjust some of the prerequisites
- Add electives

Current sequence:

151, “Fundamentals of computer science I”
152, “Fundamentals of computer science II”
201, “Memory management, data representation, and formal methods”

Proposed sequence:

151, “Functional problem solving”
161, “Imperative problem solving”
207, “Object-oriented programming”

Current description of 151:

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.

Proposed description for 151:

A lab-based introduction to basic ideas of computer science, including recursion, abstraction, **scope and binding, modularity**, the design and analysis of algorithms, **and the fundamentals of** programming in a high-level, functional language.

Adjustments in the topics covered:

Out: objects, files, assignment, stacks and queues

In: **libraries, media**

Current description of 201:

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.

Proposed description for 161:

A continuation of Computer Science 151, bringing in some concepts more closely tied to the architecture of computers, compilers, and operating systems, such as macro processing, compilation and linking, **pointers** and **memory** management, data representation, and software development tools. Additional topics include **assertions and invariants**, data abstraction, linked data structures, an introduction to the use of the GNU/Linux operating system, and programming in a low-level, imperative language.

Adjustments in the topics covered:

Out: formal verification methods (except for assertions, invariants, and pre- and post-conditions)

In: more on the GNU/Linux toolset, state and assignment, macros, flags and masks

Why place the imperative model ahead of the object-oriented model?

- (1) Statements within a method body in an OO language operate on the state of the object just as statements in an imperative language operate on the state of the machine.
- (2) Understanding pointers and other low-level imperative ideas makes it easier to understand the behavior of objects and object references in an OO language.
- (3) Majors can meet the prerequisite for 211 and 213 sooner.
- (4) The OO model is inherently more complicated.
- (5) Moving C into the second course makes it easier for physics majors to take it.

Current description of 152:

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.

Proposed description for 207:

An introduction to the ideas and practices that characterize the **object-oriented** model of computation: message passing, information hiding, classes and interfaces, inheritance, polymorphism, and reflection. The course also includes **data structures** and the associated algorithms, packages and libraries, exceptions, and the use of an integrated software-development environment.

- Rearrange the introductory courses
- **Allow more options in the major requirements**
- Adjust some of the prerequisites
- Add electives

A new course, Computer Science 225, would be an alternative to 223 (“Software design”), focussing on Web software development and deployment.

Computer Science 362, “Compilers,” would become an alternative to Computer Science 302, “Programming language concepts.”

These four courses would be offered in a four-semester rotation: 362 in the fall of (say) even-numbered years, 223 in the following (odd-numbered) spring, 225 in the following (odd-numbered) fall, and 302 in the following (even-numbered) spring.

- Rearrange the introductory courses
- **Allow more options in the major requirements**
- Adjust some of the prerequisites
- Add electives

Current requirements:

Mathematics 218
Computer Science 152 or 153
Computer Science 201
Computer Science 211 or 213
Computer Science 223 or 362
Computer Science 301
Computer Science 302
Computer Science 341

Proposed requirements:

Computer Science 151 or 153
Computer Science 161
Computer Science 207 or 153
Computer Science 211 or 213
Computer Science 223 or 225
Computer Science 301
Computer Science 302 or 362
Computer Science 341

- Rearrange the introductory courses
- Allow more options in the major requirements
- **Adjust some of the prerequisites**
- Add electives

Current prerequisites:

152: 151
 201: 152 or 153
 205: 151 or 153, LIN 114
 211: 201
 213: 201
 223: 152 or 153
 261: 152 or 153
 301: 152 or 153, MAT 218
 302: 301
 341: 152 or 153, MAT 218
 362: 201
 364: 201

Proposed prerequisites (tentative)

161: 151 or 153
 205: 151 or 153, LIN 114
 207: 161
 211: 161
 213: 161
 223: 207 or 153
 225: 207 or 153
 261: 161
 301: 207 or 153, MAT 218
 302: 207 or 153
 341: 161, MAT 218
 362: 207 or 153
 364: 207 or 153

- Rearrange the introductory courses
- Allow more options in the major requirements
- Adjust some of the prerequisites
- **Add electives**

Current electives:

205, “Computational linguistics”
 261, “Artificial intelligence”
 364, “Computer networks”
 Internships, guided readings,
 independent projects, +2s

Possible electives (depending
 on interest and staffing):

Human-computer interaction
 Value-sensitive design
 Computer graphics
 E-commerce
 Advanced topics in algorithms
 Numerical methods / scientific
 computing
 Robotics
 Animation / computer games
 Parallel and distributed
 computing
 Cryptography / information
 security

This presentation was created with OpenOffice.org Impress.

It is available on MathLAN (in Open Document Format) at
</home/stone/service/department/2007/curriculum-proposal.odp>.