Session on Curricular Challenges and Responses

Panel Slides from Henry M. Walker, Grinnell College

Three Interconnected Challenges

- Meeting the needs of both majors and non-majors
- Incorporating paradigms/multiple views of problem solving within the curriculum
- Leveling the playing field for beginning students who arrive with varying backgrounds

Coordinated Responses

Majors versus non-majors

- At Grinnell, students do not have to declare majors until the end of their second year
  - Cannot distinguish between majors and non-majors in introductory CS courses
  - CS1 focuses on algorithmic problem solving and functional programming---a good start for any student, whether a potential major or not
  - Introductory CS courses meet needs of both majors and non-majors
- In practice, large fraction of potential majors (perhaps 2/3) did not consider CS before entering Grinnell
  - Eventual majors captivated by CS1, CS2, ...
- Grinnell has a separate non-majors course
  - Mostly taken by juniors and seniors
  - Focus on algorithmic thinking and computing topics the common citizen should know about.
  - In practice, this is a fine service course, but has relatively low demand.

Coordinated Responses

Multiple views of problem solving/Leveling the playing field

- At Grinnell, we highlight multiple views of problem solving early.
  - The college faculty understand multiple perspectives fit well with study of the liberal arts.
  - Few other programs within the college care what problem-solving is done in CS
    - Thus, CS program is free to do what it wants.
- CS1: functional problem solving (supported by Scheme)
  - New to almost all students, so helps neutralize different backgrounds among incoming students
  - Often includes an application theme (e.g., image processing, data analytics)
- CS2: imperative problem solving (supported by C)
  - Includes low-level computing elements (e.g., run-time stack, data representation, dynamic memory)
  - Includes discussion of linked lists, ADTs, stacks, queues
  - Often includes an application theme (e.g., control of robots)
- CS3: object-oriented problem solving (supported by Java)
  - Classes, objects, interfaces, inheritance, polymorphism, etc.
  - Common standard algorithms and data structures (e.g., hash tables, some graphs)
  - Discussion of efficiency (e.g., Big-O, storage considerations)

Coordinated Responses

Additional notes

- All courses are lab-based with heavy use of pair programming/collaboration
- Since CS1 new to all, most students start there
- Students with more background (e.g., from high school) may skip CS2 and/or CS3 (not CS1)
- Students with strong AP CS scores
    - 4 credits earned (credit separated from placement)
    - Likely start in CS1 (but may or may not skip a later course)
- With this multi-paradigm approach early, little need for standard upper-level programming-paradigms course.
- Approach seems to be well received by wide range of diverse populations
    - seems to be influenced by numerous factors (e.g., multi-paradigm, lab-based, work in pairs, collaboration---building a sense of community, application themes, etc.)