CS2136: Paradigms of Computation

Class 01: Course Overview
History of Programming Languages

Copyright 2001 - 2004 Michael J. Ciaraldi and David Finkel
Overview

- Introduction of the course staff
- Purpose of the course
- What will be covered
- What won’t be covered
- Logistics
- Policies
Course Staff

Professor 
David Finkel  
- dfinkel@wpi.edu  
- x5416  
- FL 231  
- http://www.cs.wpi.edu/~dfinkel/
Course Staff II

z Teaching Assistants
  y Mahim Singh
    x mahim@cs.wpi.edu
  y Song Wang
    x songwang@cs.wpi.edu
Course Staff III

z Senior Assistants
  y Arno Hautala
    x arno@wpi.edu
  y Iavor Trifonov
    x trifonov@wpi.edu
Email addresses

- The TAs and SAs (and me):
  - cs2136-ta@cs.wpi.edu

- The entire class (plus the TAs, SAs, and me):
  - cs2136-all@cs.wpi.edu

Please use only for course-related topics.
Texts

“Required”

  • Available in bookstore or electronically at http://www.bruceeckel.com/

  • Available in bookstore.

For more background

Louden, Kenneth C. Programming Languages: Principles and Practice. 1993. PWS.
Purpose

z Examine different approaches to solving problems using computers.
  y Object-oriented programming
  y Logic programming
  y Functional programming

z An excuse to learn languages
  y Java
  y Prolog
Purpose II

CS2135:
- Covered functional programming.
- Covered Scheme.
- Other than that, what’s the difference?
"If the only tool you have is a hammer, every problem looks like a nail."—Maslow.
Paradigms affect...

- how you see the problem
- how you express the problem
- how you go about solving the problem
- the solutions you consider
- the solutions you can implement
What Will Be Covered

- Different paradigms
  - And what makes them different
- Different languages
  - And how they match the paradigms
- Some history and motivation
  - Why do it this way?
- Some interesting techniques
  - E.g. multithreading, GUIs
What Will Not Be Covered

- All the details of some languages
- Languages for the sake of languages
What I Assume You Know

- Programming in some high-level language.
- Something about object-oriented programming.
Logistics

- Meet here four times a week.
- Check email regularly.
- Course Web site:
  - [http://www.cs.wpi.edu/~dfinkel/Courses/cs2136.html](http://www.cs.wpi.edu/~dfinkel/Courses/cs2136.html)
- Slides will be posted on the course Web site
  - In ppt and pdf; if you have trouble viewing the pdf, download and open with your pdf reader.
  - (printing, bringing, annotating suggested).
- You are responsible for any announcements made in class or made by email to the class mailing list.
Software

z Get access to a Prolog interpreter.
  y Already on the CCC Unix machines.
  y For MS-Windows
    x Download SWI-Prolog.

z Get access to a Java compiler.
  y Already on the CCC Unix machines.
  y For MS-Windows
    x Download Sun Java Development Kit (JDK) or
    x Borrow Visual Studio (including J++) from Gordon Library.
Policies

- Grading
- Quizzes
- Homework / Projects
- Academic Honesty
Grading

z Approximate Weighting
  y Quizzes: 5%
  y Homework / Projects: 45%
  y Exams: 50%
Quizzes

- Cannot be made up.
- Beginning of class.
- Randomly scheduled.
- Mostly factual.
- Cover the day’s reading and/or previous day’s class.
Homework / Projects

z Grading standards.

y Program must compile and start running without error. Otherwise, no credit.

y Proper design, formatting, and documentation.
Due 8:50 am on due date.

- Penalty per class day or fraction: 10% of maximum.
  - There will be a final due date after which assignments will not be accepted.

- Turn in using turnin.
  - Instructions will be provided

- Schedule shows approximate dates - may need to be modified
Exams

- Exam 1 will cover logic programming and Prolog
- Exam 2 will cover object-oriented programming and Java
- Schedule shows approximate dates; may need to be modified depending on class progress.
Academic Honesty

- Read and follow the policy.
- Motivation.
- Basic rule:
  - You may discuss approaches with other students
  - You may not see the code of another student or allow another student to see your code.
History of Programming Languages
Two Great Books

Why have programming languages?
Pre-History

- c. 1840: Charles Babbage designed the Analytical Engine
  - Programmed by punch cards
  - Never built
  - Might have worked.
- Ada Augusta, Countess of Lovelace, wrote programs for it.
1940s

z 1943: Harvard Mark I: First electromechanical computer.
   y Howard Aiken (IBM), Grace Hopper (Navy)

z 1946: ENIAC: First electronic computer.
   y Presper Eckert, John Mauchly (U. Penn.)

z Programming was manual, with switches and cables.

z Harvard Mark II: First bug.
The First Bug

0800 Andan started

1000

13° WC (032) HP-MC 2.13076u15 conA 1.41593u053 (2)

Relays 6,2 in 033 found special speed test

1100 Started Cosine Tape (sine check)

1525 Started Multi-Adder Test

1545 Relay #70 Panel F (moth) in relay

First actual case of bug being found.

1700 Changed startd.
1950s

c. 1950: First stored-program computers built.

John von Neumann (Princeton) pointed out that programs could be stored in memory like data.

Early ’50s: Programming in machine language.

1952: The first assemblers.

Univac
1950s II

1954: FORTRAN

- “Formula Translation”
- First high-level language
  - Farther from machine, closer to problem
- John Backus (IBM)
- Great for manipulating numbers
- Call by value, call by reference
- Later, call by value-replace
- Global variables using common and named common.
1950s III

z 1958-1960: Algol 58, Algol 60

y “Algorithmic Language”

y First languages with structure
  x Program structure
  x Control structure

y Recursive; uses a stack

y Call by name

y Peter Naur
1950s IV

1959-1960: COBOL

- “Common Business Oriented Language”
- Based on English
  - ADD A TO B GIVING C.
- Instigated by the Department of Defense
- Grace Murray Hopper
- No parameters or local variables
- Great facilities for manipulating text and numbers
1950s V

1959: LISP
- “List Processing”
- McCarthy (MIT)
- First functional language
- Notation based on lambda calculus
- Interpreted
- Programs and data represented interchangeably as lists of lists and atoms
- Self-modifying code possible
Next Time

More history