CS 2SC3 and SE 2S03 Fall 2009

00 Preliminaries

William M. Farmer

Department of Computing and Software
McMaster University

10 September 2009

Instructor

- Dr. William M. Farmer
- Office: ITB 163
- Extension: 27039
- E-mail: `wmfarmer@mcmaster.ca`
- Web: `http://imps.mcmaster.ca/wmfarmer/`
- Office hours: M 9:30–10:30, T 16:30–17:30

Teaching Assistants

- TAs:
  1. Pouya Larjani (`larjanp@mcmaster.ca`)
  2. Han Yin (Chris) Zhang (`zhanghy2@mcmaster.ca`)
  3. Rebecca Dreezer (`dreezerj@mcmaster.ca`)
  4. Kwadwo Amoako Sakyi (`sakyika@mcmaster.ca`)
- TA duties:
  - Conduct weekly tutorials
  - Answer questions concerning the course material
  - Provide assistance with programming assignments
  - Mark programming assignments
  - Help mark midterm test and final exam
- TA office hours: TBA

Mission

Imperative programming is the oldest, and still one of the most popular, programming paradigms. The mission of the course is to teach students the underlying principles of imperative programming and basic data structures. By the end of the course the student should:

1. Understand the imperative programming paradigm.
2. Be able to program in the imperative style in Objective Caml (OCaml).
3. Be able to program in the imperative style in C.
4. Have a working knowledge of Unix-style operating systems, command-line interfaces, and version control systems such as Subversion.
5. Have a sense of what are the professional responsibilities of computing professionals.

Mechanics

- Lectures: MW 8:30–9:20, F 10:30–11:20 in MDCL 1105
- Tutorials: One per week
- Course web site:

  `http://imps.mcmaster.ca/courses/SE-2S03-09/`

- Course outline: Read it closely!
- E-LearnMac (ELM) page:

  `http://elm.mcmaster.ca/`

- i>clicker system:
  - Each student is required to obtain an i>clicker remote
- The class will pick a class representative who will serve as a liaison between the students and the instructor

TextBooks

1. E. Chailloux, P. Manoury, and B. Pagano, Developing Applications with Objective Caml, O'Reilly France, 2000. English translation available for free at `http://caml.inria.fr/pub/docs/oreilly-book/`.
2. K. N. King, C Programming: A Modern Approach, Second Edition, Norton, 2008. ISBN 978-0-393-97950-3.

Work Plan

- Lectures given by the instructor
- Tutorials conducted by the TAs
- Six programming assignments done individually
  - Assignments will be submitted electronically
  - Each student is required to keep a log book
  - Students will use Unix, command-line interfaces, and version control software (Subversion)
- Short quiz each Friday using the i>clicker system
- Midterm test during class time on Friday, October 30
- 3-hour final exam on the date scheduled by the University

Log Book

- Each student is required to keep a detailed, up-to-date log book
- Format: Text file in which the entries are listed chronologically with dates and times
- Should include:
  - Steps performed on the programming assignments
  - Sources of information, consultations with instructors, teaching assistants, and fellow students
  - Successful and failed experiments
  - Discovered errors
  - Lessons learned
- A copy of your log book must be submitted with each programming assignment

Academic Dishonesty

- Students are expected to exhibit honesty and use ethical behavior in all aspects of the learning process
- Academic dishonesty consists of misrepresentation by deception or by other fraudulent means
- Academic dishonesty includes:
  - Plagiarism
  - Copying
  - Improper collaboration
- Academic dishonesty can result in serious consequences
- Your work must be your own. Plagiarism and copying will not be tolerated!
- Students may be asked to defend their written work orally

Discrimination

The Faculty of Engineering is concerned with ensuring an environment that is free of all adverse discrimination. If there is a problem, that cannot be resolved by discussion among the persons concerned, individuals are reminded that they should contact their Department Chair and the Human Rights and Equity Services (HRES) office as soon as possible.

Missed Work

- A Missed Work Form (MWF) is required for missed work.
- To get an MWF, the student must go to Engineering Services in JHE A214 with appropriate official documentation
- MWFs will not be accepted for missed lectures

Course Modifications

The instructor and university reserve the right to modify elements of the course during the term. The university may change the dates and deadlines for any or all courses in extreme circumstances. If either type of modification becomes necessary, reasonable notice and communication with the students will be given with explanation and the opportunity to comment on changes. It is the responsibility of the student to check their McMaster e-mail and course web sites weekly during the term and to note any changes.

Other Policy Statements (1/3)

1. Each student is strongly urged to submit a completed Questionnaire and Biographical Sketch which is available on the course web site.
2. Significant study and reading outside of class is required.
3. The student is expected to ask questions during class.
4. You may want to discuss the assignments with your fellow students. If you do that, you must record a summary of your discussions in your log book including a list of all those with whom you had discussions and a description of what information you received. It is part of your professional responsibility to give credit to all who have contributed to your work.

Other Policy Statements (2/3)

5. If there is a problem with the marking of an assignment, the student should first discuss the problem with the TA who marked the assignment. Assignment marks will only be changed if the problem was reported within two weeks of the date that the assignment was returned.
6. A student may use his or her texts and notes during the midterm test and final exam but not during the quizzes.
7. Programming assignments may not be submitted late and the quizzes and midterm test may not be taken later without prior approval from the instructor.
8. No electronic devices (calculators, cell phones, etc.) may be used on tests or brought into testing rooms. Hats are also not allowed in testing rooms.

Other Policy Statements (3/3)

9. The instructor reserves the right to require a deferred final exam to be oral.
10. Suggestions on how to improve the course and the instructor's teaching methods are always welcomed.

Marking Scheme

Quizzes (10)	20%
Programming exercises (6)	20%
Midterm test	20%
Final exam	40%
Total	**100%**

Notes: A student's final score will be reduced by one half point for each missed lecture. However, there is no penalty for the first five missed lectures for which participation is recorded. A student who answers less than half of the questions asked during a lecture via i>clickers will be considered as having "missed" the lecture. Missed lectures will be excused only in highly exceptional cases.

Syllabus

00 Preliminaries

01 Programming Languages

02 Functional Programming

03 Imperative Programming

04 Basic Data Structures

05 Case Study: C Programming Language

Software is Everywhere

- In homes
- In offices
- In travel
- In communication
- In entertainment
- In manufacturing plants and machine tools
- In banks and financial markets
- In hospitals and medical devices
- In research
- In the devices we hold in our hands
- In engineering and science

Software is Transforming our World

- Software is truly everywhere!
- Software is revolutionizing almost all human endeavors
- Software is giving humans greater power, greater reach
- The Internet is bringing humanity closer together
- Software is radically changing engineering and science

Software Development is Full of Challenges

- The digital nature of software makes it fragile
- Software applications are often exceedingly complex
  - Software systems are some of the most complex artifacts ever created by humans
  - Computer science and software engineering are to a very large extent disciplines for managing complexity
- Software technology is constantly changing
- Many software systems are mission-critical, safety-critical, security-critical, cost-critical, etc.
  - Critical software must not fail!

Kinds of Software

- There are two major kinds of software:
  1. Software that runs on a general-purpose computing platform such as a server or personal computer
  2. Software inside an embedded system that controls a physical device
- Examples of embedded systems:
  - Nuclear power plants
  - Automobiles
  - Programmable household devices
  - Aircraft
  - MP3 players
  - Radio Frequency Identification (RFID) tags
- Embedded systems are rapidly appearing everywhere
- The developers of software for an embedded system need to understand both the software and the physical device

Example: Cell Phones

- A cell phone is a portable, computer-controlled telecommunication device
  - A cell phone is also a computing platform
  - Security issues are a major concern
- A cell phone is an excellent example of a embedded system
  - Cell phone software is special-purpose software
  - Cell phones contain 600,000 to 10,000,000 lines of computer code!
  - The developers of cell phone software need a deep understanding of how cell phones work

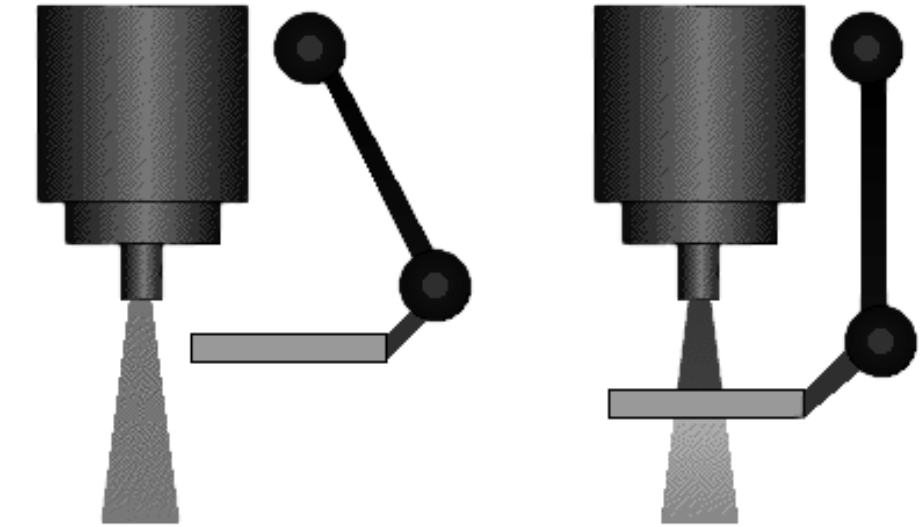
The Great Software Crisis

- Software is often exceedingly difficult and expensive to develop
- Software projects often fail
- Software systems often fail
- Software developers do not guarantee their work
- There are not enough people who have an adequate understanding of how to specify, design, implement, test, and maintain software
- Although there are many widely accepted software development principles, there is not an accepted, best-practice system for developing software
- Software development tools are inadequate
- There is no accepted system for certifying software

Example: Therac-25 (1/4)

- The Therac-25 was a radiation therapy machine for treating cancer
  - Produced by the Atomic Energy of Canada Limited (AECL)
  - Controlled by software
- How it worked:
  - Provided both electron beam and X-ray treatment
  - The machine produced low- to high-energy electron beams
  - X-rays were produced by rotating a target into the path of a high energy electron beam
- Used in several clinics across North America

Example: Therac-25 (2/4)



Electron Mode

X-Ray Mode

Example: Therac-25 (3/4)

- In six separate incidents in the 1980s, Therac-25 machines delivered overdoses of radiation causing severe physical damage or even death to the patients being treated
  - The second incident, which took place in Hamilton, resulted in an administration of 13,000–17,000 rads of radiation (200 rads is a regular treatment and 1000 rads can be fatal)
  - Three patients ultimately died from radiation poisoning
- What went wrong:
  - Software failed to detect that the target was not in place
  - Software failed to detect that the patient was receiving radiation
  - Software failed to prevent the patient from receiving an overdose of radiation

Example: Therac-25 (4/4)

Causes of the failure:

- Inadequate software design
- Inadequate software development process
  - Coding and testing done by only one person
  - No independent review of the computer code
  - Inadequate documentation of error codes
  - Poor testing procedures
- Software was ignored during reliability modeling
- No hardware interlocks to prevent the delivery of high-energy electron beams when the target was not in place
- Software was not certified in any reasonable manner

The Great Gulf

- Engineers do not sufficiently understand or care about software
  - Many of the basic principles of software design and development are largely unknown to engineers
  - Engineers often do not appreciate the challenges and dangers inherent in software for embedded systems
- Software developers lack engineering training and professionalism
  - There is an entrenched culture of producing software without any guarantee whatsoever
  - There is no system for certifying either software or software developers
  - Most software developers lack the engineering background needed to produce software for embedded systems

Challenges and Opportunities for Engineering

- Challenges:
  - Engineers need to design embedded systems that have safe, correct, high-quality software
  - Software engineers and computer scientists need to produce software they can guarantee
- Opportunities:
  - Software tools can greatly enhance the capabilities of engineers
  - Software can greatly increase the effectiveness of the devices engineers design

Example: Automotive Camshafts (1/2)

- A camshaft is a mechanical device for controlling the opening and closing of the intake and exhaust valves of a car engine
- Engine performance depends on valve timing and thus on the shape of the cams on a camshaft
- Problem: Optimal valve timing varies according to the speed of the engine
  - A standard camshaft system can only be optimized for one engine speed!
- Solutions:
  - Honda's VTEC (Variable Valve Timing and Lift Electronic Control) system that enables a engine to have multiple camshafts
  - Ferrari's camshaft system with three-dimensional cams
  - A set of computer-controlled valve lifters

Example: Automotive Camshafts (2/2)

The set of computer-controlled valve lifters illustrates the challenges and opportunities that software offers to engineering

- Challenges:
  - The development team needs a deep understanding of both engine mechanics and software design
  - The software must work correctly and must be able to handle all likely kinds of failure
  - System failure can be catastrophic!
- Opportunities:
  - Valve timing can be programmed with almost unlimited flexibility
  - The valve timing can be optimized for each engine speed
  - Other factors, such as engine temperature, can be inputs to the control software for the valve lifters

Bridging the Gulf

- The engineering and software development professions must bridge the gulf between engineering and software
- McMaster's undergraduate programs are designed to contribute to this bridge:
  - Software Engineering
    - One of the first three undergraduate software engineering programs accredited in Canada (2001)
    - Specialty programs in Game Design and Embedded Systems
  - Mechatronics Engineering
  - Computer Science
- The construction of the bridge has been started, but your generation of engineers will need to finish it!