We use cookies on this website. If you continue to access our website, we'll assume that you consent to receiving cookies in accordance with our **Privacy Policy (http://www.cs.ox.ac.uk/privacy-policy.html)**.

Imperative Programming I: 2016-2017

Lecturer	**Geraint Jones (/people/Geraint.Jones)**
Degrees	**Preliminary Examinations (/teaching/csp/PreliminaryExaminations) — Computer Science and Philosophy (/teaching/csp)**
	Preliminary Examinations (/teaching/bacompsci/PreliminaryExaminations) — Computer Science (/teaching/bacompsci)
	Preliminary Examinations (/teaching/mcs/PreliminaryExaminations) — Mathematics and Computer Science (/teaching/mcs)
Term	Hilary Term 2017 (16 lectures)

Overview

This course applies lessons that have been learnt in Functional Programming to the design of programs written in an imperative style. By studying a sequence of programming examples, each a useful software tool in its own right, students learn to construct programs in a systematic way, structuring them as a collection of modules with well-defined interfaces.

The course introduces the idea of loop invariants for understanding and reasoning about loops. The course also introduces the idea of modularising larger programs, capturing the functionality of a component of the program using an abstract mathematical specification, and describing formally the relationship between that specification and the implementation.

Through lab exercises, students learn to create, debug and maintain programs of a non-trivial but moderate size.

Learning outcomes

After studying this course, undergraduates will be able to:

1. Translate basic functional idioms into imperative ones.
2. Design simple loops, using invariants to explain why they work correctly.
3. Use subroutines and modules to structure more complex programs.
4. Specify a module as an abstract datatype, and formalise the relationship between that specification and an implementation.
5. Design simple data structures.
6. Understand the imperative implementation of some common algorithms.

Synopsis

- Basic imperative programming constructs: assignments, conditionals, procedures and loops. Comparison of imperative and functional programming. Examples.

- Method of invariants: correctness rules for **while** loops; proof of termination. Examples including summing an array, slow and fast exponentiation. Unit testing; debugging.
- Examples: string comparison, printing numbers in decimal.

- Binary search.
- Quicksort.
- Programming with abstract datatypes.
- Objects and classes as modules; specification; data abstraction.
- Reference-linked data structures: linked lists, binary trees.

Syllabus

Imperative programming constructs, with informal treatment of invariants. Procedures and modules; their use in the design of large programs; specification and implementation of abstract datatypes. Data structures: arrays, reference-linked data structures. Basic tools for program development. Case studies in design of medium-sized programs.

Reading list

There is no set text for the course, in the sense of a book that is followed by the lectures. I shall be keeping in mind a book about Oberon:

- Reiser & Wirth, Programming in Oberon, Addison--Wesley, 1992.

That book is out of print, but there is **a scanned version (http://spivey.oriel.ox.ac.uk/wiki/files/rd/ProgInOberonWR.pdf)** available.

Mike Spivey's Oberon compiler implements the Oberon-2 dialect, with a few minor differences that are explained in the Lab Manual. The language itself is described in **the Oberon--2 report (ftp://ftp.ethoberon.ethz.ch/Oberon/OberonV4/Docu/Oberon2.Report.ps)** from **ETHZ (https://www.ethz.ch/en.html)** (**local copy in PDF (http://spivey.oriel.ox.ac.uk/wiki/files/rd/o2report.pdf)**).

There are many adequate treatments of the use of logic and invariants in the development of imperative programs; one reasonably pitched one is

- Gries, The Science of Programming, Springer, 1981.

Useful additional cultural reading, recommended for reading after the course, perhaps during the Easter vacation:

- Jon Bentley, *Programming Pearls*, Dorling Kindersley, 2006.

Calendars (/calendars.html)
Internal (http://intranet.cs.ox.ac.uk/home2/)
RSS Feeds (/rssfeeds.html)
Sitemap (/sitemap.html)
Privacy & Cookies (/privacy-policy.html)

(https://www.linkedin.com/company/department-of-computer-science-university-of-oxford)

(https://twitter.com/CompSciOxford)

(https://www.facebook.com/DepartmentOfComputerScienceUniversityOfOxford)

(https://www.flickr.com/photos/computerscienceoxford/)

© University of Oxford 2016
/teaching/courses/imperativeprogramming1/index.html