CS 59: Principles of Programming Languages

Sergey Bratus, Fall 2025

2A, room TBD

Objectives: To explore features and abstractions of programming languages that make them "modern" and motivate their ongoing development. To understand how these features are implemented in actual interpreters, compilers, and virtual machines. To get a taste for different styles of programming: functional, object-oriented, continuation-passing, "monadic",[1] and others. To look under the hood of language abstractions.

In practical terms, this course should give you some idea why languages like LISP, Scheme, and Haskell were invented, why languages like Perl, Python, and Ruby transformed the industry, and why leading companies continue developing new languages such as Swift, Go, and Rust.

Required background: The course assumes that you wrote relatively complex programs in a language like C, C++, Java, Python, or Ruby, and have a good understanding of recursive data structures such as lists and trees. For this reason, CS 10 is a requirement and CS 30 is recommended. Having programmed in LISP, Scheme, ML, Haskell, or other functional languages will be helpful but not required.

You will be expected to either have or quickly develop an understanding of how a language like C is compiled down to machine code, and how that machine code executes on the actual CPU. For this reason, CS 51 is recommended, but if you are familiar with x86 or ARM machine code from other sources, or took a class in compilers, that will be sufficient.

Expect to do at least as much or more time looking at what your code turns into at the lower layers ("under the hood") as you'd spend writing the code.

The X-hour: As a rule, we will **not** be using the X-hour for lectures, but may use it for optional meetings (e.g., to catch up on a background subject). Each use of the X-hour will be announced on the class mailing list.

Office Hours: My office is 216 ECSC. Please email me for appointments.

Grades: 20% homework, 40% midterm, 40% final project.

Homework will be given, but its primary purpose is for you to check your understanding of the material and fluency with new languages. The midterm will be take-home, open-everything, and hard; do not expect to develop the required fluency overnight!

Software and Hardware: Linux or MacOS preferred. We will make heavy use of the Unix shell and tools. Your mileage with Windows *will* vary.

Course Topics:

- How imperative languages such as C compile and run. How to look for language artifacts in compiled machine code.
- Understanding the runtime: the Application Binary Interface (ABI); compilation, linking, and loading of libraries; dynamic linking.
- Scripting languages. Internals of a Ruby interpreter.
- Functional programming: why aspire to program without "state" and "side effects" (or loops). Functions as "first-class objects".
- Recursion and induction.
- A taste of LISP(s) and programs as their own fundamental data types.

[1] The mathematical foundations of monads, functors, and other concepts that Haskell and other cutting-edge programming languages explore are beyond the scope of this course; still, we'll touch on some ideas behind them.

- Lambda calculus, and what it has to do with programming. Other models of computation.
- Closures: the idea and the implementation.
- Continuations and the continuation-passing style.
- Types and type-checking.
- Polymorphism and generic programming. How overloading and inheritance work in C++ and in interpreted languages.
- Strongly typed languages and type inference.
- A taste of ML and Haskell: functional, strongly typed languages with polymorphism.
- Algebraic data types and a new style of programming: pattern-matching, "monadic" handling of errors.
- Laziness and its puzzles.
- Extra: An idea of why "Propositions are Types" and "Types are Programs".

Textbooks:

– *Concepts in Programming Languages* by John C. Mitchel. **This is our main textbook, covering most of the theory.** It is pricey if you buy it new, but used copies are available fairly cheaply, from Amazon and other online stores. Some sites also offer electronic copies—please exercise judgment and caution when dealing with unknown websites.

– *On Lisp* by Paul Graham, available as free download from `http://www.paulgraham.com/onlisp.html`

– *Ruby Under a Microscope: An Illustrated Guide to Ruby Internals* by Pat Shaughnessy

– *Real World Haskell* by Bryan O'Sullivan, Don Stewart, and John Goerzen, freely available online at `http://book.realworldhaskell.org/read/`

– We will also use chapters from the free *Structure and Interpretation of Computer Programs* (2nd ed.) by Harold Abelson and Gerald Jay Sussman with Julie Sussman `https://web.mit.edu/6.001/6.037/sicp.pdf`

Supplemental on Lisp:

– *Common Lisp the Language* by Guy Steele (2nd ed.), freely available at `https://www.cs.cmu.edu/Groups/AI/html/cltl/cltl2.html`

– *Practical Common Lisp* by Peter Seibel, freely available at `http://www.gigamonkeys.com/book/`

– The book I like best: *ANSI Common LISP* by Paul Graham. Unfortunately, not available freely from the authors, but worth owning if you are interested in LISP.

Supplemental on Haskell:

– *Learn You a Haskell for Great Good* by Miran Lipovača. Freely available online at `http://learnyouahaskell.com/`

More supplemental reading will be suggested during the course.

Required to be included in every syllabus:

Students requesting disability-related accommodations and services for this course are required to register with Student Accessibility Services (SAS; Apply for Services webpage; *student.accessibility.services@dartmouth.edu*; 1-603-646-9900) and to request that an accommodation email be sent to me in advance of the need for an accommodation. Then, students should schedule a follow-up meeting with me to determine relevant details such as what role SAS or its Testing Center may play in accommodation implementation. This process works best for everyone when completed as early in the quarter as possible. If students have questions about whether they are eligible for accommodations or have concerns about the implementation of their accommodations, they should contact the SAS office. All inquiries and discussions will remain confidential.