CS 381: Programming Language Fundamentals
Course Introduction

Winter 2021
Q: Why study programming languages?

A: Languages are at the ♥ of computer science!

A: Good programming languages really matter!
Outline

Why study programming languages?
  Languages are at the heart of computer science
  Good languages really matter

How to study programming languages

Course logistics
What is computer science?

Computer science is no more about computers than astronomy is about telescopes.

—Edsger Dijkstra

Computer Science = the science of computation
What is computation?

Computation = **systematic transformation of representation**

- **Systematic**: according to a fixed plan
- **Transformation**: process that has a changing effect
- **Representation**: abstraction that encodes particular features

Languages play a central role:

- The “fixed plan” is an **algorithm**, which is described in a **language**
- The “representation” is **data**, which is also often described in a **language**
What about software engineering?

Science vs. Engineering

Science: tries to understand and explain
Engineering: applies science to build stuff

Science  Engineering
physics    structural engineering, …
chemistry  chemical engineering, …
“computing” software engineering, …

Both are part of “computer science”
Central role of PL in CS

PL supports both aspects of CS:

• to understand and explain (science)
  we need **languages** to describe and reason about computations for ourselves

• to build cool stuff (engineering)
  we need **languages** to describe computations for a computer to execute
Outline

Why study programming languages?
  Languages are at the heart of computer science
  Good languages really matter

How to study programming languages

Course logistics
Why good languages matter: preventing bugs

Good languages can help prevent bugs

• Mars Climate Orbiter failure, 1998
  • caused by mismatched units between ground and spacecraft
  • lost $327.6 million + years of effort

• Heartbleed bug in SSL, 2012–2014
  • caused by missing bounds check
  • huge violations of privacy, including 4.5 million medical records
  • estimated $500 million in damage

• Steam’s Linux client deletes root, 2015
  • caused by silent failure of a directory lookup operation
  • offending line commented by “Scary!”… :-/
Large-scale software systems are complex!

Good languages can help us **manage this complexity**

- “Structured programming”, 1950–1960s
  - problem: “spaghetti code” caused by GOTOs
  - solution: subroutines, conditionals, loops

- Rust programming language, Mozilla, 2010s
  - problem: managing memory in low-level, concurrent systems code
  - solution: ownership system
Why good languages matter: medium of thought

The languages we use …
- influence our **perceptions**
- guide and support our **reasoning**
- enable and shape our **communication**

- What problems do we see? How do we reason about and discuss them?
- How do we develop, express, and share solutions?

*By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race.*

—Alfred North Whitehead via Kenneth Iverson’s ACM Turing Award Lecture, “Notation as a Tool of Thought”
Example: Positional number system

In the 13th century, this is how numbers were represented in Europe:

$$\text{MMCDXXXI} \div \text{XVII} = ? \quad :-(\$$

…even basic arithmetic is hard!

Fibonacci popularized the Hindu-Arabic notation

- didn’t just make arithmetic much more convenient …
- completely changed the way people thought about numbers, revolutionizing European mathematics
Example: Symbolic logic

For over 2000 years the European study of logic focused on syllogisms

Every philosopher is mortal.
Aristotle is a philosopher.
Therefore, Aristotle is mortal.

Only 256 possible forms … field solved!

A couple of notational innovations in the 19th century cracked it wide open

- George Boole – Boolean algebra
- Gottlob Frege – Beggriffsschrift (symbolic predicate logic)
Example: Feynman diagrams

Interactions of subatomic particles lead to brain-melting equations

• reasoning about interactions requires complex math
• high overhead to communicating problems and solutions

Only a handful of people can do this stuff!

In 1948, Richard Feynman introduced a visual language for representing interactions

Raises level of abstraction

• eliminates incidental complexity (math)
• focus on essential complexity (interactions)
• supports communication, collaboration (undergrads can do it)
Domain-specific languages

\[ F = ma \]
\[ E = mc^2 \]
## Impact of programs and languages

<table>
<thead>
<tr>
<th>If you can …</th>
<th>then you can …</th>
</tr>
</thead>
<tbody>
<tr>
<td>run a program</td>
<td>do something faster and more reliably</td>
</tr>
<tr>
<td>write a program</td>
<td>empower others to do new things</td>
</tr>
<tr>
<td>create a language</td>
<td>unlock potential for solving harder problems and bring the power of computing to new domains</td>
</tr>
</tbody>
</table>

Each level is a **multiplier** for potential impact!
Bottom-up strategy: try out a bunch of languages

Goal: learn new ways of thinking by programming in very different languages
Top-down strategy: programming language concepts

Focus on how to define programming languages

Learn how to:

- define the **structure** of programs in a language
- define the **meaning** of programs in a language
- identify the **features** that are common to many languages
Summary of our strategy

**Introduce two new ways of thinking** (bottom up)
1. functional programming (Haskell) – lots of practice
2. logic programming (Prolog) – last couple of weeks

**Introduce programming language concepts** (top down)
1. reduce languages to their essential features
2. define their abstract syntax
3. define their semantics
4. consider important design decisions

We use Haskell as a metalanguage for describing these concepts!
**Role of metalanguages**

**Metalanguage**: a language to define the structure and meaning of another language!

In this course:
- grammars
- English
- Haskell
- Prolog

Important metalanguage not covered:
- mathematics
- inference rules
Outline

Why study programming languages?
- Languages are at the heart of computer science
- Good languages really matter

How to study programming languages

Course logistics