Welcome to CS 152: Programming Paradigms

Michael McThrow

San Jose State University
Computer Science Department

August 23, 2021

Table of Contents

Table of Contents

Welcome!

Welcome to CS 152: Programming Paradigms!

In this course, we will be studying different paradigms of programming, with an emphasis on functional and logic programming.

We will not only be programming in different programming languages, but we will also be writing interpreters to learn about how programming languages are implemented.

We will cover theory and practice.

The Essence of Programming

Programming is about expressing computations. Programming languages are the media in which we express these computations.

One of the main goals of this course is to learn different perspectives, or paradigms, of programming.

Why learn different paradigms? "A change in perspective is worth 80 IQ points," Alan Kay, creator of the Smalltalk programming language, which is one of the programming languages we will be studying in this course.

Table of Contents

Online Delivery

- CS 152 will be conducted entirely online this semester.
- There will be no on-campus meetings this semester.
- All assignments and exams will be delivered electronically.
- I will be hosting virtual office hours on Zoom each Monday from 6:15pm to 7:15pm, excluding holidays (September 6).
- I can also meet with students individually by appointment.

Class Structure

1. (First five minutes of class) Short Review of Previous Class
2. Main Lesson
3. (Last 15 minutes of class) Question and Answer Session

Etiquette

- Please do assigned readings before the lecture; see syllabus for more information.
- Please keep your microphones muted except when I'm taking questions or while we are having a class discussion.
- Please be respectful in all communication in this course.
- Please keep up with all communication in this course, including email messages and Canvas postings.

Class Attendance

- Attendance is not mandatory in my course.
- I will be recording the course lectures for the benefit of students who can't attend class during the scheduled time.
- However, any topic discussed during the lectures may show up on exams, so please come to class or watch the recorded videos.
- Also, the question-and-answer sessions may be very valuable.

Prerequisite

- CS 151 (Object-Oriented Design) or CMPE 135 (Object-Oriented Analysis and Design)
- Either class must be passed with a grade of C- or better.
- I will be enforcing this prerequisite. Please send me proof that you've met this prerequisite by August 30, 2021 in order to not be dropped from the course.

Textbooks

- *Structure and Interpretation of Computer Programs, 2nd Edition* by Ableson, Sussman, and Sussman (1996)
- *Teach Yourself Scheme in Fixnum Days* by Sitaram (2015)
- *The Art of Prolog, Second Edition: Advanced Programming Techniques* by Sterling and Shapiro (1993)
- *Smalltalk-80: The Language and Its Implementation* by Goldberg and Robson (1983)

All of these textbooks are available online for free. Please refer to the syllabus for links to these books.

Assignments

- There will be five projects assigned that consist of substantial programming exercises, including implementing two programming language interpreters (one for Scheme and one for Prolog).
- There will be two take-home exams: a midterm and a final exam.

Late Work Policy

- Deadlines will be at 11:59:59pm Pacific Daylight Time (Pacific Standard Time starting November 7).
- 10% late penalty for projects turned in up to three days after the deadline.
- No projects will be accepted after the three-day extension period.
- No extensions for take-home exams except under extenuating circumstances; please contact me ASAP if you encounter such circumstances.

Grading

- Projects: 60%
- Midterm: 20%
- Final Exam: 20%

Please refer to syllabus for more information about how final grades will be determined.

Academic Integrity

All exams and non-group projects are to be done individually.

Group projects are to be done in pairs (two students), though students may optionally work alone (which counts as a group of one). Groups may not collaborate with each other.

Studying in groups and sharing already-graded work with each other is OK.

Sharing exam questions or posting them online before exams have been graded is prohibited.

Key takeaway: Academic integrity is vital in order for online education to be effective.

Academic Integrity

All instances of academic dishonesty will be reported per University Policy F15-7. Please see syllabus for a link to this policy.

Table of Contents

Procedural Paradigm

All of you have experience with the procedural paradigm of programming, which emphasizes step-by-step instructions that are generally[1] executed sequentially.

[1] I say "generally" because there's the fact of out-of-order instruction execution on modern CPUs.

Procedural Prime Number Finder Example in Java

```
private boolean isPrime(int n) {
    if (n < 2) return false;
    else if (n == 2) return true;
    else {
        for (int i = 2; i < n; i++)
            if (n % i == 0) return false;
        return true;
    }
}

public List<int> findPrimes(int n) {
    ArrayList<int> primes = new ArrayList<int>();
    for (int i = 1; i <= n; i++)
        if (isPrime(i)) primes.add(i);
    return primes;
}
```

Some of you may know other procedural, imperative languages such as C, C++, JavaScript, and Python where you can solve this problem in a similar way.

But what if I showed you a different way of solving this problem using a different programming paradigm?

Functional Prime Number Finder Example in Scheme

```
(define (prime? n)
  (cond ((< n 2) #f)
        ((= n 2) #t)
        (else (not (divisible-between-1-and-n? n 2)))))

(define (divisible-between-1-and-n? n x)
  (cond ((= n x) #f)
        ((= (remainder n x) 0) #t)
        (else (divisible-between-1-and-n? n (+ x 1)))))

(define (find-primes n)
  (cond ((< n 2) ())
        ((= n 2) '(2))
        ((prime? n) (cons n (find-primes (- n 1))))
        (else (find-primes (- n 1)))))
```

That was a little different.

Now, here is something a bit more different....

Logic Programming Prime Number Finder Example in Prolog

```
div1AndN(N,X) :- 0 is N mod X, !.
div1AndN(N,X) :- N > X+1, div1AndN(N, X+1).

prime(2).
prime(N) :- not(div1AndN(N, 2)).

findPrimes(N,L):-findall(X,(between(2,N,X),prime(X)),L).
```

This is cool and all, but why learn these alternative ways of programming? Isn't it true that programming languages don't matter?

Turing Completeness

Yes, it is true that any Turing-complete programming language is able to perform the same computations as any other Turing-complete programming language.

Turing Completeness

Yes, it is true that any Turing-complete programming language is able to perform the same computations as any other Turing-complete programming language.

However, this does not mean that all programming languages have the same runtime characteristics, nor does this mean that all programming languages have the same degree of expressiveness.

Reasons for Learning Different Programming Paradigms

Reasons for Learning Different Programming Paradigms

- Certain paradigms are more appropriate in different situations.

Reasons for Learning Different Programming Paradigms

- Certain paradigms are more appropriate in different situations.
- Certain problems are more naturally expressible under certain programming paradigms.

Reasons for Learning Different Programming Paradigms

- Certain paradigms are more appropriate in different situations.
- Certain problems are more naturally expressible under certain programming paradigms.
- Learning different ways of solving problems aids in general problem-solving skills.

Reasons for Learning Different Programming Paradigms

- Certain paradigms are more appropriate in different situations.
- Certain problems are more naturally expressible under certain programming paradigms.
- Learning different ways of solving problems aids in general problem-solving skills.
- Some programming languages are multi-paradigm, allowing users to mix-and-match different paradigms in the same language.

Reasons for Learning Different Programming Paradigms

- Certain paradigms are more appropriate in different situations.
- Certain problems are more naturally expressible under certain programming paradigms.
- Learning different ways of solving problems aids in general problem-solving skills.
- Some programming languages are multi-paradigm, allowing users to mix-and-match different paradigms in the same language.
- Helps develop taste in programming languages.

On Developing Taste

Even within the same paradigm, there can be large differences between programming languages.

On Developing Taste

Even within the same paradigm, there can be large differences between programming languages.

- Why do C and C++ have pointers but Java does not?

On Developing Taste

Even within the same paradigm, there can be large differences between programming languages.

- Why do C and C++ have pointers but Java does not?
- Why do some people disparage PHP and JavaScript, yet why are these languages so commonly used?

On Developing Taste

Even within the same paradigm, there can be large differences between programming languages.

- Why do C and C++ have pointers but Java does not?
- Why do some people disparage PHP and JavaScript, yet why are these languages so commonly used?
- Why do some programming language researchers criticize C?

On Developing Taste

Even within the same paradigm, there can be large differences between programming languages.

- Why do C and C++ have pointers but Java does not?
- Why do some people disparage PHP and JavaScript, yet why are these languages so commonly used?
- Why do some programming language researchers criticize C?
- Static typing vs. dynamic typing?

On Developing Taste

Even within the same paradigm, there can be large differences between programming languages.

- Why do C and C++ have pointers but Java does not?
- Why do some people disparage PHP and JavaScript, yet why are these languages so commonly used?
- Why do some programming language researchers criticize C?
- Static typing vs. dynamic typing?
- Should programming languages encourage safety or promote freedom?

On Developing Taste

Even within the same paradigm, there can be large differences between programming languages.

- Why do C and C++ have pointers but Java does not?
- Why do some people disparage PHP and JavaScript, yet why are these languages so commonly used?
- Why do some programming language researchers criticize C?
- Static typing vs. dynamic typing?
- Should programming languages encourage safety or promote freedom?

Studying the design differences between programming languages helps develop taste.

Is learning different programming language paradigms relevant in industry?

Lisp and Prolog in Industry

- Reddit was originally written in Lisp, though it was later rewritten in Python (see this article by the late Aaron Swartz).
- Grammarly's core logic is written in Common Lisp (see "Running Lisp in Production" by Vsevolod Dyomkin here).
- Lisp and Prolog were the main languages used in artificial intelligence before the machine learning revolution began.

Plus, industry trends change over time.

Programming Language Implementation

Also, not only will we be covering programming language paradigms in this course, we will also be covering the implementation of programming languages, with an emphasis on *interpreters*.

Overview of Course Schedule

- Overview of Course (Week 1)
- Procedural and Structured Programming (Week 1)
- Syntax and Semantics (Weeks 2 and 3)
- Functional Programming, Scheme, Interpretation, and Compilation (Weeks 4-7)
- Type Systems and More Functional Programming (Weeks 7 and 8)
- Midterm (Week 9)
- Logic Programming and Prolog (Weeks 10-13)
- Object-Oriented Programming Revisited (Weeks 13-15)
- Final Exam (Week 16)

A Sneak Preview of The Next Lecture

- The next lecture will be centered around procedural programming.
- I will be covering the history of important procedural languages and the development of *structured programming*, which had an enormous impact on programming.
- Reading: "Go To Statement Considered Harmful" by Edsger W. Dijkstra (yes, the same Dijkstra who developed Dijkstra's Algorithm).

Questions?