Lecture Outline

- Introduction: the rules!
  - Strangest Proglang ever

- Programming language spectrum
- Why study programming languages?

- Compilation and interpretation

Read: Scott Chapter 1

Introduction

Course webpage

https://www.cs.rpi.edu/~milanova/csci4430

- Schedule, Notes, Reading
  - Schedule, lecture slides, assigned reading, and homework links

- Submittit
  - Homework submission and grades (Rainbow grades)
  - Discussion forum/Announcements

- Please, submit your TIME ZONE!

Syllabus

- Topic: Language pragmatics
- Canvas: https://www.cs.rpi.edu/~milanova/csci4430/syllabus.html
- Topics, outcomes, policies, and grading
- 2 midterm exams and a final exam: 47%
- 7 homework assignments: 42%
- 8 quizzes: 8%

- 3% office hours check-in

Required textbook


Recommended textbook

Introduction

- Homework is due at 2pm on the due date
- Submit typed homework as a PDF electronically in Submitty
- Submit programming homework in Submitty for autograding
- Homework, including submission instructions, will be posted at
  - https://www.cs.rpi.edu/~milanova/csci4430/schedule.html

- 6 late days in total
- 2 late days at most per homework
- Extensions only with a formal excuse note from your class dean. See syllabus for details.

- Quiz or exam makeup will be arranged only after we have received an excuse note from your class dean. See syllabus for details.

- Office hours
  - We plan for ample office hours on Mondays, Wednesdays and Thursdays
  - Instructor office hours right after class
  - We’ll require weekly office hour “check-ins” starting at week 3 for at least 10 weeks
  - TA and mentor office hours via Submitty queues

- Asking questions
  - First, go to Submitty forum
  - Do not post code on forum
  - You cannot post code to any website
  - Second, go to office hours
  - Sessions are individual, run through Submitty queues, so you can
  - Third (last resort): proglang-help@cs.lists.rpi.edu goes to instructors
  - We will not be answering questions coming in late at night or in the morning on day HW is due.
### Other Notes

- **Submitty forum**
  - Announcements – check regularly
  - Ask all non-personal questions on the forum
  - Check out prior messages before you post a question – the answer is probably already there

- **Mailing list** prolang@cs.lists.rpi.edu (instructors)
  - Personal questions (extensions, grade disputes, etc.)
  - Unsolicited debugging emails to instructors or mailing list will likely go unanswered!

### Academic Integrity

- In short, do not copy and do not post your solutions/code on public forums or repos
- Excessive similarities between homework submissions will be considered cheating and handled accordingly
- I trust you. Submitty has advanced plagiarism detection tools that course stuff runs regularly

### How to Study

- **A 15 min presentation from the ALAC:**
  - [https://mediasite.mms.rpi.edu/Mediasite5/Play/3c69d5096dc5494eadcaba2b9c99189f1d](https://mediasite.mms.rpi.edu/Mediasite5/Play/3c69d5096dc5494eadcaba2b9c99189f1d)

### How to Study

- **Debugging and homework help in office hours**
  - Instructor, TA and mentor office hours will be finalized by the beginning of next week

### Course Topics

- Programming language syntax: Scanning and parsing
- Programming language semantics: Attribute grammars
- Naming, binding and scoping
- Data abstraction and types
- Control abstraction and parameter passing
- Concurrency

- Logic-oriented language: Prolog
- Functional languages: Scheme and Haskell
- Imperative languages
  - An object-oriented language: Java
  - A dynamic language: Python
Course Topics

- Schedule at www.cs.rpi.edu/~milanova/csci4430/schedule.html
- Lists major and minor topics
- Homework links, dates and due dates
- Quiz and exam schedule

Lecture Outline

- Introduction to the course
- Programming language spectrum
  - Why study programming languages?
  - Compilation and interpretation

The Programming Language Spectrum

- Imperative languages
  - Von Neumann languages: Fortran, C, ...
  - Object-oriented languages: Java, C++, Smalltalk, ...
  - Dynamic languages: Perl, Python, PHP, ...

- Declarative languages
  - Functional languages: Scheme/Lisp, ML, Haskell
  - Logic languages: Prolog
  - There are other declarative languages: e.g., dataflow languages

The Programming Language Spectrum

- Imperative languages
  - Evolved from the von Neumann Architecture
  - Variable
  - Assignment Statement

The Programming Language Spectrum

- FORTRAN was invented in mid-1950
- John Backus, the inventor of FORTRAN, wrote the following paper in 1979: “Can programming be liberated from the von Neumann style? A functional style and its algebra of programs”
  - Problems with imperative languages
    - Difficult to understand programs
    - Difficult to reason about correctness of programs
The Programming Language Spectrum

- **Functional Programming**
  - Main alternative to imperative programming
  - Lisp/Scheme, ML/OCaml, Haskell
  - Program consists of function definitions + evaluation expr
    \[(\text{fun } 3 \ (\text{fun } 2 \ (\text{fun } 1 \ \text{data})))\]
    \[(\text{fun } 3 \ (\text{fun } 2 \ \text{data}2))\]
    \[(\text{fun } 3 \ \text{data}3)\]
  - Execution is a sequence of function applications (i.e., reductions)

- **Logic Programming**
  - Perform queries against knowledge base
  - Prolog, Datalog, SQL

An Example: Inner Product

**Inner product in FORTRAN:**

1. \(C := 0;\)
2. for \(I := 1\) step 1 until \(N\) do
3. \(C := C + a[I]^*b[I];\)

- Illustrates state-transition semantics

**Inner product in FP:**

Def \(\text{IP} = (\text{Insert } +) \circ (\text{ApplyToAll } \ast) \circ \text{Transpose}\)

\(\text{IP} <<1,2,3>,<6,5,4>>\) is
\((\text{Insert } +) ((\text{ApplyToAll } \ast) (\text{Transpose} <<1,2,3>,<6,5,4>>))\)
\((\text{Insert } +) ((\text{ApplyToAll } \ast) <<1,6>,<2,5>,<3,4>>))\)
\((\text{Insert } +) <6,10,12>\)

- Illustrates reduction (applicative) semantics

Why Study Programming Languages

**Goal of the course:** learn to analyze programming languages

- What are the questions we ask when facing a new programming language
  - Helps learn new languages, choose the right language for a problem, understand language features, **design better languages**!
Compilation

Scanner (lexical analysis)
- Character stream
- Token stream

Parser (syntax analysis)
- Parse tree

Semantic analysis and intermediate code generation
- Abstract syntax tree
- Intermediate form

Intermediate form (three-address code)
- tmp1 = id3 * 60.0
- tmp2 = id2 + tmp1
- id1 = tmp2

Machine-independent code improvement
- modified intermediate form
- target language

Code generation
- Improved intermediate form
- Target language

Machine-dependent code improvement
- modified

Compilation

position = initial + rate * 60;

Symbol table
1. position, ...
2. initial, ...
3. rate, ...

Parser
- Scanner

Semantic analysis and intermediate code generation

Abstract Syntax Tree
- <id,1>
- <id,2>
- <id,3>
- 60.0

Intermediate form
- tmp1 = id3 * 60.0
- tmp2 = id2 + tmp1
- id1 = tmp2

Semantic analysis and intermediate code generation

Compilation

Interpretation

- e.g. BASIC
  - REM COMMENT
  - LET X = 5
  - LET Y = X
  - PRINT X
  - LET Z = X
  - PRINT Z

Hybrid Interpretation

- e.g. Java byte code
- e.g. Java Virtual Machine (JVM)
- Also Perl...

Source program
- Interpreter
- Input data
- Results
The End