Lecture Outline

- Introduction: the rules
  - We are back in the classroom!

- Programming language spectrum

- Why study programming languages?

- Compilation

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

- Submitty
  
  Homework and quiz submission and grades (Rainbow grades)
  
  Discussion forum
Introduction

- Required textbook
Introduction

- Recommended textbook

Aho and Ullman 2020 Turing Award citation:

For fundamental algorithms and theory underlying programming language implementation and for synthesizing these results and those of others in their highly influential books, which educated generations of computer scientists.
Introduction

- Syllabus
  https://www.cs.rpi.edu/~milanova/csci4430/syllabus.html
  Topics, outcomes, policies, and grading
- 2 midterm exams and a final exam: 50%
- 7 homework assignments: 42%
- 8 quizzes: 8%
- 1% attendance and participation
Lectures are live here in DCC 308 Tuesdays and Fridays 2:00–4:00pm
  - If you are unable to attend due to medical reasons, please notify us by September 9th

Fall 2020 pre-recorded lectures are available at https://mediasite.mms.rpi.edu/Mediasite5/Channel/programming_languages

PDF notes: https://www.cs.rpi.edu/~milanova/csci4430/schedule.html
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
Introduction

- Homework is due at 2pm on the due date
- 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.
Introduction

- 9 quizzes during scheduled class hours
  - Quiz dates are marked in Schedule
- Will cover material of previous weeks
- Quiz will show on Submitty at 2pm, 10-15min
  - Multiple choice and short answer, upload text file
  - Work in groups is encouraged (4-5 people)
    - Physical attendance not required but encouraged
  - Do not post questions or discussion on public sites/channels!
Exams will be in person in DCC 308

- If unable to attend due to medical reasons, please notify us by September 9th

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

- We plan for ample office hours on Mondays, Wednesdays and Thursdays. TBD.

- Instructor office hours
  - Mondays 12:30-2:30pm in Lally 314. Please wear a mask
  - Immediately after class on Tuesdays and Fridays
Other Notes

- 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

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

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

- In short, **do not copy and do not post solutions or 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

- Read textbook chapter in advance of lecture
  - Chapters are announced on Schedule page
- Read/listen lecture and read textbook chapter immediately after class
  - Lecture pdfs will be available shortly before class
- Solve exercises in lectures
- Form study groups
- ASK QUESTIONS – in class, on forum
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
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

![Diagram of the von Neumann computer architecture](image)

**Figure 1.1**
The von Neumann computer architecture

Central processing unit

Input and output devices
The Programming Language Spectrum

- Imperative languages
  - Most widely popular programming style
    - FORTRAN, C, C++, C#, Java, Python, Visual BASIC, Perl, JavaScript, Ruby, etc.
  - Variable and assignment statement are central concepts
  - Program is a sequence of statements:
    \[ j := i - j; \]
    \[ k := j * l; \]
    - Execution is a sequence of transitions on memory state
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

- John Backus 1977 Turing award citation
  … contributions to the design of practical high-level programming systems, notably through his work on FORTRAN, and for seminal publication of formal procedures for the specification of programming languages.

- More history…
  - 1969: Hoare logic and program verification
  - 78-79: Enthusiasm cools, Perlis’ paper, Backus’ paper
  - 1980-ties and onward: Functional languages
The Programming Language Spectrum

- Functional Programming
  - Main alternative to imperative programming
    - Lisp/Scheme, ML/OCaml, Haskell
  - Program consists of function definitions + evaluation expr
    (fun3 (fun2 (fun1 data)))
    (fun3 (fun2 data2))
    (fun3 data3)
    data4
  - 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. \( \text{for } I := 1 \text{ step } 1 \text{ until } N \text{ do} \)
3. \( T := a[I] \times b[I]; \)
4. \( C := C + T; \)

Illustrates state-transition semantics
An Example: Inner Product

Inner product in FP:

Def IP = (Insert +) ° (ApplyToAll *) ° Transpose

IP <<1,2,3>,<6,5,4>> is
(Insert +) ((ApplyToAll *) (Transpose <<1,2,3>,<6,5,4>>))
(Insert +) ((ApplyToAll *) <<1,6>,<2,5>,<3,4>>)
(Insert +) <6,10,12>
28

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 languages
Lecture Outline

- Introduction to the course

- The programming language spectrum
- Why study programming languages

- Compilation
Compilation and Interpretation

- **Compilation**
  - Compiler
  - A “high-level” program is translated into executable machine code

- **Pure interpretation**
  - Interpreter
  - A program is translated and executed one statement at a time

- **Hybrid interpretation**
  - Both a compiler and an interpreter
  - A program is “compiled” into intermediate code; intermediate code is “interpreted”
Compilation

Scanner (lexical analysis)
- token stream

Parser (syntax analysis)
- parse tree

Semantic analysis and intermediate code generation
- abstract syntax tree and/or intermediate form

COMPILER

Machine-independent code improvement
- modified intermediate form

Code Generation
- target language (assembly)

Machine-dependent code improvement
- modified target language
Compilation

position = initial + rate * 60;

Scanner

{id,1} <=> {id,2} <+> {id,3} <*> <60>

Parser

Semantic analysis and intermediate code generation

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

Semantic analysis and intermediate code generation

\[
\begin{align*}
tmp1 &= id3 \times 60.0 \\
tmp2 &= id2 + tmp1 \\
id1 &= tmp2
\end{align*}
\]

Intermediate form (three-address code)
Compilation

Machine-independent code improvement

tmp1 = id3 * 60.0
id1 = id2 + tmp1

Code generation

movf id3, R2
mulf #60.0, R2
movf id2, R1
addf R2, R1
movf R1, id1

Improved intermediate form

Target language
Interpretation

Source program

Input data

Interpreter

Results

e.g. BASIC

REM
COMMENT
LET X = 5
LET Y = 8
PRINT X
PRINT Y
LET Z = X
PRINT Z
...

...
Hybrid Interpretation

- Source program
- Lexical analyzer
- Lexical units
- Syntax analyzer
- Parse trees
- Intermediate code generator
- Intermediate code
- Interpreter
- Input data
- Results

E.g. Java byte code

E.g. Java Virtual Machine (JVM)

Also Perl....
Compilation vs. Interpretation

Advantages of compilation?
- Faster execution

Advantages of interpretation?
- Greater flexibility
  - Dynamic code generation and execution, sandboxing
A language can be implemented using a compiler or using an interpreter.

- One can build a compiler for Lisp and one can easily build an interpreter for C or Fortran.

However, language features (determined during language design) have significant impact on “compilability” and the decision “compiler vs. interpreter.”
Next Class

- We will review regular expressions and context free grammars

- Read Chapter 2.1 and 2.2 from Scott’s book
The End