CS 326
Programming Languages, Concepts and Implementation

Instructor: Mircea Nicolescu

Introduction

Contacts

- Instructor: Dr. Mircea Nicolescu
  - E-mail: mircea@cse.unr.edu
  - Office: WPEB 413
  - Office Hours: Thursday: 10:30am-1:30pm

- Class web page:
  - http://www.cse.unr.edu/~mircea/Teaching/cs326

Assignments and Exams

- Homework assignments (40%)
  - Written exercises and small programs
  - More complex programs, in Scheme, Prolog, Java or ML
- Midterm exam (25%)
  - Closed book, closed notes
- Final exam (30%)
  - Closed book, closed notes
  - Comprehensive, although focused on the second half of the course
- Class participation (5%)

Late Submission Policy

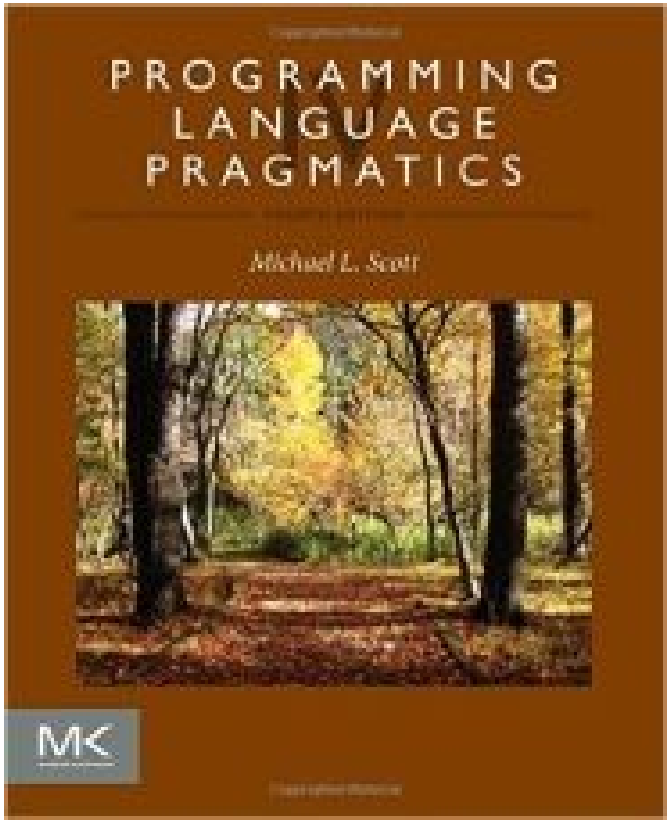
- No late assignments will be accepted

Warning:

Dates in the calendar are closer than they appear!

Required Textbook

- Programming Language Pragmatics,
  by Michael L. Scott, Morgan Kaufmann Press, 2015.



What Are We Studying?

- Programming languages characterized by:
  - Syntax – what a program looks like
  - Semantics – what a program means
  - Implementation – how a program executes
- Trade-offs between design (specifying syntax and semantics) and implementation
- Will focus on these concepts, illustrated by various languages, and not on the languages themselves

Why Study Programming Languages?

- Help you choose a language
  - Systems programming → C, Modula-3
  - Scientific computations → Fortran, Ada
  - Embedded systems → C, Ada, Modula-2
  - Symbolic computing → Lisp, Scheme, ML

- Make it easier to learn new languages
  - Some languages are similar
  - Some concepts are even more similar: iteration, recursion, abstraction

Why Study Programming Languages?

- Make better use of the languages you know
  - Improve ability to develop effective algorithms
  - Understand obscure features
  - Understand implementation costs
  - Simulate useful features in languages that do not support them
- Prepare for further study in language design and implementation (compilers)
- Help understand other system software (assemblers, linkers, debuggers)

Historic Perspective

- When did the earliest high-level languages appear?
  - Continuous evolution since the 1950s (Fortran, Cobol)
- When the Department of Defense did a survey as part of its efforts to develop Ada in the 1970s, how many languages did it find in use?
  - Over 500 languages were being used in various defense projects

Architecture Evolution

- 1950s – Large expensive mainframe computers ran single programs (Batch processing)
- 1960s – Interactive programming (time-sharing) on mainframes
- 1970s – Development of microcomputers. Early work on windows, icons, and PCs at XEROX PARC
- 1980s – Personal computer: IBM PC and Apple Macintosh. Use of windows, icons and mouse
- 1990s – Client-server computing - networking, the Internet, the World Wide Web

Spectrum of Languages

- Imperative ("how should the computer do it?")
  - Von Neumann: Fortran, Basic, Pascal, C
    - Computing via "side-effects" (modification of variables)
  - Object-oriented: Smalltalk, Eiffel, C++, Java
    - Interactions between objects, each having an *internal state* and *functions* which manage that state
- Declarative ("what should the computer do?")
  - Functional: Lisp, Scheme, ML, Haskell
    - Program ↔ application of functions from inputs to outputs
    - Inspired from *lambda-calculus* (Alonzo Church)
  - Logic, constraint-based: Prolog
    - Specify constraints / relationships, find values that satisfy them
    - Based on *propositional logic*

Why Are There So Many Languages?

- Evolution
  - We have learned better ways of doing things over time
- Socio-economic factors
  - Proprietary interests, commercial advantage
- Special purposes
- Special hardware
- Personal preference

What Makes a Language Successful?

- Easy to learn
  - Basic, Pascal
- Easy to express things
  - “Powerful”
  - Easy to use once known
  - C, Lisp
- Easy to implement
  - Small machines, limited resources
  - Basic, Forth

What Makes a Language Successful?

- Efficient compilers
  - Generate small / fast code
  - Fortran
- Backing of a powerful sponsor
  - Ada, Visual Basic
- Wide dissemination at a minimal cost
  - Pascal, Java
- Inertia
  - Fortran, Cobol

Standardization

- Is `i = 1 && 2 + 3 | 4;` legal in C?
- What is assigned to `i` if it is?

- 3 ways to answer this:
  - Read language manual (Problem: Can you find one?)
  - Read language standard (Problem: Have you ever seen it?)
  - Write a program to see what happens. (Easy to do!)

- Most do the last, but different compilers may give different answers

Standardization

- Language standards defined by:
  - ISO - International Standards Organization
  - IEEE - Institute of Electrical and Electronics Engineers
  - ANSI - American National Standards Institute
- Contentious features omitted to gain consensus
- Nothing is enforced

Standardization

- When to standardize a language:
  - If too late – many incompatible versions already exist (Fortran in 1960)
  - If too early – no experience with language (Ada in 1983 had no running compilers yet)
  - Just right – probably Pascal in 1983, although it is rapidly becoming a dead language

- Other languages:
  - C in 1988
  - LISP in 1990 – way too late
  - De facto standards: ML – one major implementation: SML
  - Smalltalk, Prolog – none

Standardization

- Internationalization:
  - How many bits to use for characters
  - Additional letters (German ä, French é)
  - Other alphabets (Cyrillic, Greek, Arabic, Chinese)

- Organization that studies this problem cannot even decide between *internationali**z**ation* and *internationali**s**ation* for its own name

- Other *internationali*ation* issues:
  - Dates, times, daylight saving time
  - Currency

Readings

- Chapter 1