Programming Language Design and Implementation

Instructor: Sandhya Dwarkadas
TAs: Bijun He and John Kramer

http://www.cs.rochester.edu/u/sandhya/csc254

What is a programming language?

A means for precisely expressing (syntax and semantics) what you want a computer to do
- User: Way of expressing algorithms
- Implementer: abstraction of machine

- Why are there so many languages?
  - Evolution
  - Socio-economic factors
  - Orientation toward special purposes
  - Diverse ideas about what is pleasant to use

What makes a language successful?

- Ease of learning (BASIC, Pascal, Scheme)
- Ease of expression (C++, Perl, Lisp)
- Ease of implementation (BASIC)
- Ease of optimization (Fortran)
- Sponsorship
- Wide dissemination (Pascal, Java)

Programming language views

- User's perspective
  - Ease of use
  - Expressive power
- Implementer's perspective
  - Ease of implementation
  - Ease of optimization

Why study programming languages?

- Help you choose a language
- Make it easier to learn a new language
- Make better use of language features
- Help you understand other systems software
- Further study in language design or implementation

Types of languages

- Imperative
  - Von Neumann (Fortran, Pascal, BASIC, C, …)
  - Object-oriented (C++/Java, Smalltalk, …)
- Declarative
  - Functional (Scheme, Lisp, ML, …)
  - Logic, constraint-based (Prolog, VisiCalc, …)

Compilation vs. Interpretation

Compiler: Translates the high-level source program into an equivalent target program for later execution

Advantage: better performance

Interpreter: Implements a virtual machine and executes program “on-the-fly”

Advantage: greater flexibility,
better diagnostics

Phases of Compilation

- Scanning (lexical analysis)
- Parsing (syntax analysis)
- Semantic analysis
- Intermediate code generation
- Optimization
- Target code generation