Programming Languages

Lecture 1: Introduction

Benjamin J. Keller
Department of Computer Science, Virginia Tech
Lecture Outline

- Preview
- History of Programming Languages
- Course topics
Programming Languages

- Representation that allows humans to communicate with a computer
- Abstraction of a machine language
- Different languages provide different abstractions
Kinds of Abstractions

- Data abstractions
  - primitive data types: numeric, characters, booleans, pointers
  - structured types: arrays, records
  - unit: support for ADTs, modules, packages, classes

- Control abstractions
  - basic: assignment, goto, sequencing
  - structured: conditionals, loops, procedures and functions
  - unit: separately compiled units, modules, packages, concurrent tasks
Defining Languages

- *Program* — sequence of symbols that specify a computation
- *Programming Language* — a set of rules that specify which sequences of symbols constitute a program, and what computation the program describes.
- *Syntax* — how program is formed
- *Semantics* — what program does
- *Pragmatics* — how language is used
Language Goals

- Languages designed for different purposes
  - Fortran — numerical computation
  - Pascal — teaching programming
  - C — low-level system programming

- Evaluate language with respect to goals of its design
Languages and Software Development

- Language design goals may conflict with large-scale system development
- Modern languages support specific design methodology/philosophy
  - functional design
  - object-based design
  - encapsulation
  - information-hiding
- Language influences how people think about programming
Computer architecture led to *imperative*

Others developed as different ways to solve problems

- *Functional* — computation as expression evaluation, popular in AI and theoretical programming language communities.
- *Logical* — computation as proving queries. AI and database query languages.
- *Object-Oriented* — computation as manipulation of/or interaction of objects.
Language Requirements

- Universal — if a problem can be solved on a computer, then can solve it in the language
- Natural (expressive) — easy to express solutions
- Implementable
- Efficient — writing, compilation, or execution
- Reliable — ability to deal with exceptional behavior
- Maintainable — easy to make changes to programs
History

- Trend from machine language toward high-level languages
- Trend from single programmer to teams of programmers
- Early Languages
More Early History


- **SNOBOL 4** — Griswold, Bell Labs, 1966. Goals: string processing with pattern matching.

- **PL/I** — committee at IBM, 1967. Attempt to combine FORTRAN, COBOL, Algol 60. Large language difficult to learn and use.
More History

- Algol 68 — Algol with “orthogonal” elements. Elegant, but hard to understand.

- Simula 67 — simulation language based on Algol 60. Basis for most object-oriented languages.

- Pascal — Wirth, ETH-Zurich, 1971. Teaching language designed to support structured programming. Simple and elegant.
Further Developments

- Module support for abstract data types: Clu, Mesa, Modula-2, Ada
- Object-oriented: Smalltalk (70’s), Eiffel, C++, Object Pascal, Java
- Functional: Scheme (80’s), ML (70’s), Miranda, Haskell
- Logical: Prolog, newer constraint languages
Fourth Generation Languages

- Languages in business applications
- Specialized packages of powerful commands with simplified syntax.
Family Tree of Important Languages

- Fortran
- Algol 60
- COBOL
- Lisp
- PL/I
- Algol 68
- C
- PL/I
- Simula
- Pascal
- Smalltalk
- Clu
- Modula 2
- Ada
- C++
- Ada 95
- Eiffel
- Oberon
- Snobol 4
- Icon
- Clu
- Prolog
- APL
- Miranda
- Icon
- ML
- Haskell
- Java
Course Topics

- Language features: modules, classes, exception handling, generic types, etc.
- Functional, object-oriented, logical and imperative paradigms.
- Support for reliable programming: abstraction, encapsulation, information hiding, polymorphism, higher-order operators.
- Support for concurrency.
- Formal definition of programming languages.
- Compilers and interpreters.
- Run-time behavior of programming languages