Programming Languages

Week 6
CS 212 – Spring 2008

Announcements

- Monday evening GBA section has been shut down
  - If you were assigned to this section, please find a different section
  - If you cannot attend a different section, please contact one of the TAs
- Monday March 3 (next week) there will be a combination help-with-project/pizza session in place of the 12:20 sections
  - Keep an eye on the website for further information

FORTRAN

- Initial version developed in 1957 by IBM

- Example code

```
C     SUM OF SQUARES
      ISUM = 0
      DO 100 I=1,10
      ISUM = ISUM + I*I
  100 CONTINUE
```

- FORTRAN introduced many of the ideas typical of programming languages
  - Assignment
  - Loops
  - Conditionals
  - Subroutines

ALGOL

- ALGOL = ALGOrithmic Language
- Developed by an international committee
- First version in 1958 (not widely used)
- Second version in 1960 (widely used)

- Sample code

```
comment Sum of squares
begin
   integer i, sum;
   for i:=1 until 10 do
      sum := sum + i*i;
end
```

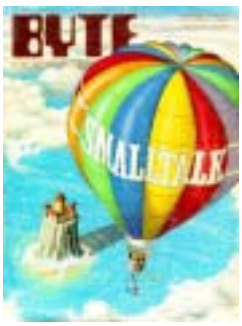
- ALGOL 60 included *recursion*
  - Pro: Makes it easy to design clear, succinct algorithms
  - Con: Too hard to implement; too inefficient

COBOL

- COBOL = COmmon Business Oriented Language
- Developed by the US government (about 1960)
  - Design was greatly influenced by Grace Hopper
- Goal: Programs should look like English
  - Idea was that *anyone* should be able to read and understand a COBOL program
- COBOL included the idea of *records* (a single data structure with multiple *fields*, each field holding a value)

Simula & Smalltalk

- These languages introduced and popularized *Object Oriented Programming* (OOP)
  - Simula was developed in Norway as a language for simulation (late 60s)
  - Smalltalk was developed at Xerox PARC in the 70s
- These languages included
  - Classes
  - Objects
  - Subclasses & Inheritance



Java

- Developed by Sun Microsystems; first released in 1995
- Java includes
  - Assignment statements, loops, conditionals from FORTRAN (but Java uses syntax from C)
  - Recursion from ALGOL
  - Fields from COBOL
  - OOP from Simula & Smalltalk

Programming Languages at Cornell

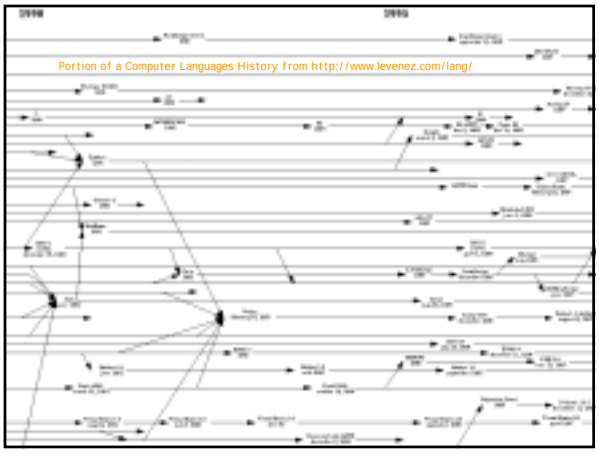
- Some of the languages used in Cornell's CS Dept
  - Java
    - 100, 211, 212
    - Many of the upper level courses
  - C, C++, C#
    - Many of the upper level courses (networks, distributed computing)
  - Matlab
    - 100M, numerical analysis courses
  - ML
    - Functional programming
    - 312, logic-related courses
  - SQL
    - Query language for databases
    - 432, database-related courses
- Fortran, C, C++, Matlab are used widely in Engineering
- SAS (originally, Statistical Analysis System) is used in various courses around campus

Some Other Programming Languages

- (from a Yahoo list)
  ABC, ActiveX, Ada, AMOS, APL, AppleScript, Assembly, awk, BASIC, BETA, C and C++, C#, Cecil, Cilk, CLU, COBOL, ColdC, cT, Curl, Delphi, Dylan, Dynace, Eiffel, Forth, Fortran, Guile, Haskell, Icon, IDL, Infer, Intercal, J, Java, JavaScript, JCL, JOVIAL, Limbo, Lisp, Logo, M - MUMPS, Magma, ML, Modula-2, Modula-3, Oberon, Obliq, Occam, OpenGL, Pascal, Perl, PL/I, Pop, PostScript, Prograph, Prolog, Python, Rexx, Ruby, SAS, Sather, Scheme, ScriptEase, SDL, Self, SETL, Smalltalk, SQL, Tcl/Tk, TOM, Verilog, VHDL, VRML, Visual, Visual Basic, Z

Some Other Programming Languages

- (from a Yahoo list)
  Ada APL Assembly awk
  BASIC C C++ CLU COBOL
  Fortran
  Java JavaScript JCL
  Lisp Logo
  Modula-3 Pascal
  PL/I PostScript Prolog Python
  SETL
  SQL Tcl/Tk VRML



So Many Languages

- Formula Translation (FORTRAN) in 1954 led to…
  - Over 2000 computer languages
- How many languages in use today?
  - Difficult to say
  - Legacy software (using outdated languages) is everywhere
- Why can't we just use one language?

Evolution of Programming Languages

- 1940s, early 1950s
  - Machine Language
    - Computation can be "mechanized"
- 1950s
  - Assembly language
    - Programs should be human-readable

Here is a language so far ahead of its time, that it was not only an improvement on its predecessors, but also on nearly all its successors.
- C. A. R. Hoare commenting on Algol 60

- Late 1950s
  - High level languages
  - Fortran
    - Compiled code can be efficient code
  - Cobol
    - Idea of structures/fields
  - Lisp
    - List as the central data structure
    - Programs as lists
  - Algol 60
    - Popularized recursion
    - Formal specification of language syntax
    - Lexical scoping rules

Evolution of Programming Languages, Cont'd

- 1960s
  - APL
    - Array programming
    - *Functional programming* style
  - PL/1
    - IBM attempt to combine best of Fortran & Cobol
  - Simula
    - Designed for doing *discrete event simulation*
    - Introduced *object-oriented programming* concepts

- 1970s
  - C
    - Designed specifically for *systems programming*
  - Smalltalk
    - First "pure" *object-oriented programming*
  - Prolog
    - First *logic programming* language
  - ML
    - *Functional programming*

Evolution of Programming Languages, Cont'd

- 1980s
  - C++
    - Combined C with object oriented programming
  - Ada
    - Designed for *systems programming*
    - Developed to use in place of 100s of languages used by US DoD
  - Perl
    - Developed (late 1980s) for text manipulation
    - Popularized in 1990s for web-site programming
  - Focus on use of *modules* for large scale software development

- 1990s
  - Programming for the internet
  - Java
    - Secure execution of remote code

- 2000s
  - Security & reliability
  - Programming for multiple processors

Classifying Programming Languages

- How its implemented
  - Compiled
  - Interpreted
- Programming paradigm
  - Procedural programming
  - Object-oriented programming
  - Functional programming
  - Logic programming
  - …
- Intended domain of use
  - General purpose
  - Systems programming
  - Scripting
  - Concurrent/distributed processes
  - Educational
  - Various other domains

Compiled vs. Interpreted

- Compiled
  - Parse code (typically create an abstract syntax tree)
  - Create machine code for entire program
- Interpreted
  - Run each statement as the statement is parsed
- Examples
  - Typically compiled: Fortran, Java, C
  - Interpreted: Matlab, Python, Logo, some versions of Basic
- Advantages/Disadvantages?
- The boundary between *compiled* and *interpreted* can be fuzzy
  - Java is compiled to produce JBC (Java Byte Code)
  - The JBC is then interpreted or JIT compiled

Some Programming Paradigms

- Procedural programming
  - Program can be broken into *procedures* (or *subroutines* or *functions*)
  - Examples: Fortran, Algol, Cobol, C, PL/1, Pascal
- Object-oriented programming
  - Program is seen as a group of cooperating *objects*
    - Ideas: encapsulation, inheritance, polymorphism
  - Examples: Smalltalk, Java, C++, C#, Python, recent Fortran, recent Cobol
- Functional programming
  - Emphasizes application of functions
  - Avoids *state*; avoids *mutable data*
  - Examples: Lisp, Clean, ML, Haskell, Scheme
- Logic programming
  - Based on use of *declarative* statements in the language of mathematical logic
  - Example: Prolog, Oz

Imperative vs. Declarative

- Imperative
  - Statements tell the computer what to do
  - Think "commands" or "recipe"
  - Examples
    - Java, C, Fortran
- Declarative
  - Describe *what* something is like rather than telling how to create it
  - Examples
    - Functional programming (Haskell)
    - Logic programming (Prolog)

Prolog Example

```
sendmore(Digits) :-
  Digits = [S,E,N,D,M,O,R,Y],     % Create variables
  Digits :: [0..9],               % Associate domains to variables
  S #\= 0,                        % Constraint: S must be different from 0
  M #\= 0,
  alldifferent(Digits),           % All elements must take different values

        1000*S + 100*E + 10*N + D    % Other problem constraints
      + 1000*M + 100*O + 10*R + E
  #= 10000*M + 1000*O + 100*N + 10*E + Y,
  labeling(Digits).               % Start the search
```

(from Wikipedia)

Languages for Different Domains

- General purpose
  - Examples: Lisp, Algol, PL/1, Scheme, Java, Python
- Systems programming
  - Emphasis on efficiency and tight control of data structures
  - Examples: C, C++, Forth, Modula-2
- Scripting
  - Examples: Unix shell, Perl, Python, Ruby, Tcl
- Concurrent/distributed processes
  - Control of multiple threads
  - Examples: Ada, Oz, Smalltalk, Java
- Educational
  - Examples: Basic, Haskell, Pascal, Python, Scheme, Smalltalk,
- Various other domains
  - Discrete event simulation: Simula
  - Web scripting: Javascript
  - Realtime applications: Ada
  - Text processing: Snobol
  - Printing: Postscript
  - …

Scripting Languages

- A *script* is a sequence of common commands made into a single program
  - Unix uses *shell scripts*
  - The *shell* is the interactive interface to Unix
  - You can combine commands from the *Unix shell* to create programs
- A *scripting language* is usually
  - Easy to learn
  - Interpreted instead of compiled
- Example scripting languages: Unix shell, Python, Perl, Tcl (Tool command language)
- Some Python code:

```
class Stack (object):
    def __init__ (self):
            self.stack = [ ]
    def put (self, item):
            self.stack.append(item)
    def get (self):
            return self.stack.pop()
    def isEmpty (self):
            return len(self.stack) == 0
```

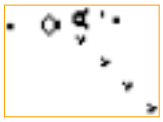
A Programming Language Controversy

- "Go To Statement Considered Harmful"
  - Edsger Dijkstra, Communications of the ACM (March 1968)
- Sparked long-running discussion on whether "go to" is necessary or desirable
  - Proponents of "go to" presented examples where code was more readable using "go to"
  - At the time
    - No *break*
    - No *continue*
    - No *exceptions*
- Led to concept of *structured programming*
  - Idea: Code is clearer if we restrict ourselves to just a few control structures
  - Loops have single entry, single exit

Just for Fun: An APL Example

```
life←{↑1 ⍵∨.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂⍵}
```

- This is the entire program for a generational update in Conway's Game of Life
  - Live cell & less than 2 live neighbors ⇒ dies of loneliness
  - Live cell & more than 3 neighbors ⇒ dies of overcrowding
  - Live cell & 2 or 3 neighbors ⇒ contentment
  - Empty cell & exactly 3 neighbors ⇒ birth



Programming Language Weirdness

- Weird languages
  - Whitespace
    - Only spaces, tabs, and newlines are significant
    - A great language for security since a program can be printed onto plain paper and stored without worrying about an adversary reading the code ☺
  - var'aq
    - Based on the grammatical structure of the Klingon language
- Weird concepts
  - Polyglot code
    - Code that is valid for multiple languages
    - Usually takes advantage of the different ways that comments are indicated in the different languages
  - Quine
    - A program whose only output is its own source code
    - Not considered valid to use the empty program

Some Advice

- Use the language that best fits your task
- Think small
  - Write little programs that test various concepts
  - Test them!
  - Comment them!
  - Build collections of these little programs
  - Reuse your own code