Programming Languages

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

- Sample code
  ```
  comment Sum of squares
  begin
  integer i, sum;
  for i:=1 until 10 do
      sum := sum + i*i;
  end
  ```
- ALGOL = ALGOrithmic Language
- Developed by an international committee
- First version in 1958 (not widely used)
- Second version in 1960 (widely used)
- 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

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

Java

- Some of the languages used in Cornell’s CS Dept
- Java
  - 100, 211, 212
- Many of the upper level courses
- C, C++, CB
- Many of the upper level courses (networks, distributed computing)
- Matlab
  - 103W, 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, C++

Some Other Programming Languages

- (from a Yahoo list)

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
  - Late 1950s
    - High level languages
      - Fortran
        - Compiled code can be efficient code
      - Cobol
        - Idea of structures/fields
        - Lists as the central data structure
        - Programs as lists
      - Algol 60
        - Popularized recursion
        - Formal specification of language syntax
        - Lexical scoping rules
- **Late 1950s**
  - High level languages
    - Fortran
    - Cobol
    - Algol 60
- **1960s**
  - APL
    - Array programming
    - Functional programming
  - 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
- **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
- **1990s**
  - Java
    - Secure execution of remote code
    - Programming for the internet
- **2000s**
  - Security & reliability
  - Programming for multiple processors

Evolution of Programming Languages, Cont’d

- **1960s**
  - 4PL
    - Array programming
    - Functional programming
    - PL/I
    - IBM attempt to combine best of Fortran & Cobol
    - Simula
      - Designed for doing discrete event simulation
      - Introduced object-oriented programming concepts

Classifying Programming Languages

- **How it’s 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
  - Run each statement as the statement is parsed

- **Interpreted**
  - Typically compiled: Fortran, Java, C
  - Interpreted: Matlab, Python, Logo, some versions of Basic

- **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 2BC (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
    - 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"
- \( \text{Examples} \)
  - Java, C, Fortran

**Declarative**
- Describe what something is like rather than telling how to create it
- \( \text{Examples} \)
  - Functional programming (Haskell)
  - Logic programming (Prolog)

---

Languages for Different Domains

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

---

Scripting Languages

- **A script is a sequence of common commands made into a single program**
- \( \text{Unix uses shell scripts} \)
- \( \text{The shell is the interactive interface to Unix} \)
- \( \text{You can combine commands from the Unix shell to create programs} \)
- **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:**
  - \( \text{class Stack(object):} \)
  - \( \text{def __init__(self)}: \)
    - self.stack = [ ]
  - \( \text{def put(self, item):} \)
    - self.stack.append(item)
  - \( \text{def get(self):} \)
    - return self.stack.pop()
  - \( \text{def isEmpty(self):} \)
    - return len(self.stack) == 0

---

A Programming Language Controversy

- \( \text{“Go To Statement Considered Harmful”} \)
  - 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 exception
  - Led to concept of structured programming
    - Ideas: Code is clearer if we restrict ourselves to just a few control structures
    - Loops have single entry, single exit

- \( \text{The concept of a "goto"} \)
- \( \text{Invented by John Backus} \)
- \( \text{Issued in 1954} \)
- \( \text{Invented in Algol} \)
- \( \text{Invented in LISP} \)
- \( \text{Invented in FORTRAN} \)

---

Prolog Example

```
sendmore(Digits) :-
    Digits = [S,E,N,D,M,O,R,Y],
    Digits :: [0..9],
    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
    + 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**
  - \( \text{Examples: Lisp, Algol, PL/I,} \) Scheme, Java, Python
- **Systems programming**
  - Emphasis on efficiency and tight control of data structures
  - \( \text{Examples: C, C++, Forth, Modula-2} \)
- **Scripting**
  - \( \text{Examples: Unix shell, Perl,} \) Python, Ruby, Tcl

- **Concurrent/distributed processes**
  - Control of multiple threads
  - \( \text{Examples: Ada, Oz, Smalltalk, Java} \)
- **Educational**
  - \( \text{Examples: Basic, Haskell,} \) Pascal, Python, Scheme, Smalltalk
- **Various other domains**
  - Discrete event simulation: Simula
  - \( \text{Web scripting: Javascript} \)
  - \( \text{Real-time applications: Ada} \)
  - \( \text{Text processing: Snobol} \)
  - \( \text{Printing: Postscript} \)
  - …

---

Scripting Languages

- **A script is a sequence of common commands made into a single program**
- \( \text{Unix uses shell scripts} \)
- \( \text{The shell is the interactive interface to Unix} \)
- \( \text{You can combine commands from the Unix shell to create programs} \)
- **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:**
  - \( \text{class Stack(object):} \)
  - \( \text{def __init__(self):} \)
    - self.stack = [ ]
  - \( \text{def put(self, item):} \)
    - self.stack.append(item)
  - \( \text{def get(self):} \)
    - return self.stack.pop()
  - \( \text{def isEmpty(self):} \)
    - return len(self.stack) == 0

---

A Programming Language Controversy

- \( \text{“Go To Statement Considered Harmful”} \)
  - 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 exception
  - Led to concept of structured programming
    - Ideas: Code is clearer if we restrict ourselves to just a few control structures
    - Loops have single entry, single exit

- \( \text{The concept of a "goto"} \)
- \( \text{Invented by John Backus} \)
- \( \text{Issued in 1954} \)
- \( \text{Invented in Algol} \)
- \( \text{Invented in LISP} \)
- \( \text{Invented in FORTRAN} \)

---

Just for Fun: An APL Example

```
life←{t∈v.a.v∧a×(t+1)<3∧∧010∧01×1∧0×1∧1∧0×0∧0×1}2
```
**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

---

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