Introduction to the study of programming languages

Course Outline
- Text
- Assignments
- Text, exam
- Schedule
- Web
  - http://www.cosc.brocku.ca/offerings/cosc3p05
- Email
  - dhughes@brocku.ca

Why Study
- Greater expressivity
- Ability to choose
- Ability to learn
- Understand implementation
- Effective use
- Evolution
Programming Domains

- Scientific
- Business
- Artificial Intelligence
- Systems
- Web programming

Comparing Languages

- Choose appropriate
- Effective use
- Design new
- Criteria
- Features

<table>
<thead>
<tr>
<th>Table 1.1</th>
<th>Language evaluation criteria and the characteristics that affect them</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRITERIA</td>
<td>SIMPLITY</td>
</tr>
<tr>
<td>Simplicity</td>
<td></td>
</tr>
<tr>
<td>Orthogonality</td>
<td></td>
</tr>
<tr>
<td>RUN TIME</td>
<td></td>
</tr>
<tr>
<td>Support for abstraction</td>
<td></td>
</tr>
<tr>
<td>Expanding</td>
<td></td>
</tr>
<tr>
<td>Type checking</td>
<td></td>
</tr>
<tr>
<td>Exception handling</td>
<td></td>
</tr>
<tr>
<td>Efficient debugging</td>
<td></td>
</tr>
</tbody>
</table>
Readability

- Program understanding
- Read more often than written
- Simplicity
  - Language subsets
  - Feature multiplicity
  - Operator overloading
  - Too simple?
- Orthogonality
  - Small number of features combined in understandable ways
  - E.g. primitive types + array & pointer
  - Exceptions to combinations
    - Can’t point to arrays
    - Functions return structs but not arrays
    - Arrays passed by reference others by value
  - Too much?
    - Algol 68

Data Types

- Appropriate to domain
  - E.g. int vs boolean
- Syntax
  - Key words vs reserved words
  - Statement delimiters
  - Overloading

Writability

- Application domain
- Simplicity
  - Too many constructs
  - Errors in combining features
- Abstraction
  - Process
  - Data
- Expressivity
  - Natural for domain
  - Extreme
    - APL
Reliability

- Program performs to specifications
- Difficult to make mistakes
- Type checking
  - Compile-time vs run-time vs typeless
- Exception handling
  - Dealing with run-time errors
  - Fail-soft vs fail-safe
- Aliasing
  - Multiple names accessing same storage location
  - Multiple pointers to same object
  - Pointers to static variables
  - Parameter and non-local referencing same variable
  - FORTRAN COMMON

Cost

- Training
- Writability
- Compilation speed
- Execution speed
- Software costs
- Reliability
- Maintenance
- Portability
- Generality

Architecture

- Von Neumann architecture
  - Imperative languages
  - Variables & assignment
  - Instruction fetch-execute cycle
- Non-von Neuman
  - Functional
    - Functional composition
  - Logic
    - Inference
- Efficiency
Design Methodologies

- Hardware costs decreasing and speed increasing
- Problem complexity increasing
  - Dealing with complexity
    - Procedural abstraction
    - Data abstraction
- Maintenance
- Methodologies
  - Structured design
  - Data-oriented design
  - Object-oriented design

Paradigms

- Imperative
  - von Neumann
    - E.g. C
- Applicative (Functional)
  - Functional composition
    - E.g. LISP, Haskell
- Logic
  - Inference
    - E.g. Prolog
- Others
  - Object-oriented
  - Scripting
  - Markup
Tradeoffs

- Criteria conflict
- Reliability vs execution speed
  - Type checking
  - Range checking
- Writability vs readability
- Expressivity vs maintainability
- Portability vs execution speed

Implementation

- High-level language vs low-level language
- Hardware design
- Machine layers
- Compilation vs Interpretation vs Hybrid
Compilation

- Translation
  - Source program to executable image
  - Slow processing fast execution
  - Early decision time
- Phases
  - Lexical analysis
  - Syntactic analysis
  - Semantic analysis
  - Code generation
- Link & load

Interpretation

- Execution via virtual machine
  - Interpreter is emulator for language-based machine
  - Execution includes analysis
    - Lexical, syntactic, semantic
  - Slower execution
  - Late decision times
  - Debugging
- Typically some translation (e.g., lexical analysis)
Hybrid

- Some translation and some emulation
- E.g. Java bytecode
- Portability
- Just-in-time compilation (JIT)