CS 350: Programming Language Design

Lecture 2

Why programming language study?

Allows better understanding of strengths and better judging of suitability for each program to a specific purpose.

You can express ideas with great efficacy, understand consequences of certain implementation features.

It may also make languages easier to learn.

Influences of Language Design

Computer architecture is the basis for how languages are developed, structured around the von Neumann structure. (CPU, ALU, Memory)

Methodologies of Program Design

New methodologies and lines of thought behind programming paradigms produce new languages that are suitable to that paradigm.

Top-Down, (Produce foundation first then more advanced functions and methods)

Bottom-Up, (Produce advanced functions then foundation)[Large amount of fault regression for functions and testing, major structure issues are not discovered until late]

OOP, (Use objects to mimic real-world things, gives balanced focus on data and functions)

Two basic Paradigms

Imperative (What is done and how, with a focus on how) C++

Declarative (Focus on what is to be done, but less focus on how) SQL

Domains of Programming

Science – Large computations (FORTRAN)

Business – Produce reports, use decimals, numbers, and chars (COBOL)

AI – Symbols rather tha numbers manipulated, Linked Lists (LISP)

System Programming – Need efficiency because of breadth of control and usage (C)

Web Software – Eclectic collection of languages markup (HTML), scripting (PHP & JS)

Language Evaluation

Readability – How can the program be read and understood

Writability – how easy is it to pick up and start writing code

Reliability – how well does it do it’s job and how frequently does it fail?

Cost – the cost to use and learn and effects on systems

Readability

Ease of maintenance is based on readability

Characteristics which affect this

* Simplicity
  + Manageable feature set (Not too many ways to do things)
  + Minimal feature multiplicity (few ways to do one thing)
  + Minimal operator overloading
* Orthogonality
  + Small set of primitive constructs which can be used in conjunction with a small number of different permutation but with each permutation being appropriate for syntax
  + Refer to coordinate set
* Data Types
  + Adequate predefined data types (having integers, Boolean, char, etc.)
* Syntax Design
  + Words used for forming compound statements.
  + Meaningful keyword (look at Linux and Unix commands) [Sudo? It means Super User Do]

Writability

* JavaFx vs. HTML for designing a GUI
* Simplicity and Orthogonality for keywords and methods
  + Few ways of combining limited number of primitives for simplicity
  + Expressivity, a set of convenient ways of specifying operations
    - Such a For loops for counting loops more simply

Reliability

Programs can be said to be reliable if it performs to its specification under all conditions.

Related characteristics

* Type checking – testing for type errors
* Exception handling – intercept run-time errors and take corrective measures
* Aliasing – different names to the same memory cell

A language that does not support natural ways of expressing an algorithm will require the use of unnatural approaches, and hence reduce reliability

Cost

* Training programmers
  + How long does it take (Simplicity and Orthogonality)
* Writing programs
  + Closeness to applications
* Reliability
  + Critical apps -> high
  + Non-Crit. -> lose future business or lawsuits.
* Maintaining programs
  + Done by different programmers than initial, readability is important.

Other criteria for evaluating languages

* Portability
  + Does a program written in C work on Mac just as well as Windows?
    - How would Java and Ruby compare?
* Generality
  + The applicability to a wide range of applications
* Definition
  + The completeness and precision of the language’s official definition

Language Design Trade-Offs

* Reliability vs. Cost
  + Java demands all references to array elements be checked for proper indexing which increases execution cost
* Readability vs. Writability
  + APL provides strong operators allowing complex computations to be written in a compact program but at the cost of poor readability
* Writability vs. Reliability
  + C++ pointers are powerful and flexible but unreliable